E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Implementación de una arquitectura de creación y reproducción de contenido MPEG-DASH

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Pablo Catalán Galar

Mikel Sagues García

Pamplona, 29 de enero de 2021

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

En el presente proyecto se realizará un análisis del estándar MPEG-DASH, su funcionamiento y las partes en las que se divide, así como la implementación de una arquitectura MPEG-DASH completa que permita crear, transmitir y reproducir contenido en streaming cuya calidad se adapte al estado de la conexión.

Para ello, primero se estudiarán las partes esenciales de un contenido codificado en DASH para conocer las ventajas y el alcance de este estándar. Posteriormente, se examinarán algunos de los programas más utilizados para la creación y la reproducción de contenido DASH.

Con este proyecto se pretende conocer más a fondo esta herramienta, muy útil y a la orden del día, teniendo en cuenta el crecimiento del sector audiovisual y la adaptación de las plataformas de streaming para satisfacer la demanda de los clientes.

Finalmente, se establecen unas líneas de continuación que podrían servir a otros alumnos para mejorar o ampliar esta investigación.

Palabras clave: MPEG-DASH, Segmento, Representation, Adaptation Set, Period, Manifiesto, H.264

Índice de contenidos

Lista de acrónimos

ABR Adaptive Bitrate. Es un algoritmo que decide qué contenido descargar en base a las condiciones de la red.

API Application Programming Interfaces. Es un conjunto de rutinas o métodos que ofrece una biblioteca para ser utilizado por otro software.

CDN Content Delivery Network. Es una red superpuesta de ordenadores que contienen copias de datos para maximizar el ancho de banda y facilitar el acceso.

DASH Dynamic Adaptive Streaming over HTTP. Es un protocolo desarrollado por MPEG y utilizado en el streaming multimedia para adaptar la calidad de los contenidos al estado de la conexión de la red.

DRM Digital Rights Management. Son las tecnologías de control de acceso para limitar el acceso a medios o dispositivos sin autorización.

GOP Group of Pictures. Es una estructura de imágenes ordenadas dentro de un flujo codificado de vídeo y limitada por frames de un tipo concreto de tal forma que facilitan su decodificación aportando más o menos información.

HLS HTTP Live Streaming. Es un protocolo desarrollado por Apple y utilizado en el streaming multimedia para adaptar la calidad de los contenidos al estado de la conexión de la red.

HTML Hypertext Markup Language. Es un lenguaje de marcado para la elaboración de páginas web.

HTTP Hypertext Transfer Protocol. Es un protocolo de comunicación que permite transferencias de información en Internet mediante hipertexto.

IEC International Electrotechnical Commission. Es una organización de normalización en los campos eléctrico y electrónico y otras tecnologías relacionadas.

ISO International Organization for Standardization. Es una organización para la creación de estándares internacionales.

MPD Media Presentation Description. Es un archivo que anuncia y describe el contenido multimedia al que hace referencia. También se le conoce como manifiesto.

OTT Over the Top. Es un servicio libre de transmisión de vídeo, audio u otros contenidos que no depende de operadores tradicionales para su control o su distribución.

RAM Random Access Memory. Es la memoria de trabajo de un ordenador para la mayor parte del software. Allí se cargan las instrucciones que ejecuta el procesador.

SAP Stream Access Point. Es un punto concreto a partir del cual un reproductor puede comenzar a decodificar la información sin tomar datos anteriores a ese punto.

SDK Software Development Kit. Es un conjunto de herramientas de desarrollo de software que permite crear una aplicación para un sistema concreto. Es una API creada para permitir el uso de cierto lenguaje que suele incluir códigos de ejemplo y notas técnicas.

TCP Transmission Control Protocol. Es un protocolo de HTTP que intercambia información de control y observa la red y los recursos del emisor y el receptor para garantizar una comunicación fiable y sin errores ni pérdidas.

URI Uniform Resource Identifier. Es una cadena de caracteres que identifica un recurso de una forma unívoca para poder acceder a él.

URL Uniform Resource Locator. Es un URI cuya dirección apunta a recursos de una red que pueden variar con el paso del tiempo.

XML Extensible Markup Language. Es un lenguaje de marcado que permite definir y almacenar datos de forma legible.

1. Introducción

1.1. Motivación

Desde 2019, el consumo de streaming supera el 60 % del tráfico total de Internet [1]. En septiembre de 2020, Netflix contaba ya con más de 182 millones de suscriptores (son más de 203 millones a principios de 2021[2]) y HBO había superado la cifra de los 140 millones de usuarios [3].

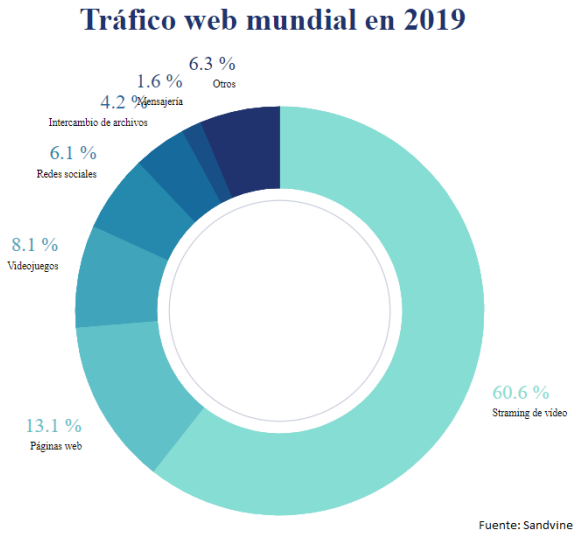

Figura 1: Tráfico web mundial en 2019

Internet es una plataforma de transporte con pérdidas que ocasiona deterioros de la calidad; y el ancho de banda y la memoria de almacenamiento de los dispositivos son limitados. Además, algunos estudios dicen que mucha gente no disfruta el contenido al completo. Concretamente, el 20-25 % de los espectadores verán menos de 10 segundos de un vídeo de YouTube u otros servicios [4]. La tecnología avanza para cubrir las nuevas necesidades que surgen con el paso de los años y la evolución de las preferencias.

En algunas ocasiones, viendo una película o un episodio de alguna serie, es posible haber notado una pérdida considerable de la calidad de vídeo. El contenido sigue reproduciéndose con esta calidad objetivamente pobre durante un tiempo indeterminado, normalmente corto, antes de volver a verse con una calidad que el usuario pueda considerar aceptable. Aunque pueda parecer un evento indeseable, es la manera que tienen hoy las plataformas de evitar pausas en la reproducción si los datos se descargan en nuestro dispositivo a una velocidad más baja de lo necesario debido a una mala conectividad. Es un avance considerable frente a las limitaciones anteriores. YouTube, además de adaptarse a la velocidad de la conexión, permite elegir la resolución de entre una lista de opciones disponibles.

Todo esto es posible gracias a MPEG-DASH, un estándar que ya utilizan Netflix [5], YouTube [6] y Prime Video [7], entre otros servicios OTT.

1.2. Objetivo

El objetivo de este proyecto es implementar una estuctura completa de creación, transmisión y reproducción de contenido codificado en MPEG-DASH.

Para ello, primero se examinarán los elementos principales del estándar y las secciones en las que se divide un contenido codificado en DASH para conocer así su funcionamiento. Entender la utilidad de cada sección es esencial para comenzar la aplicación de sus funciones.

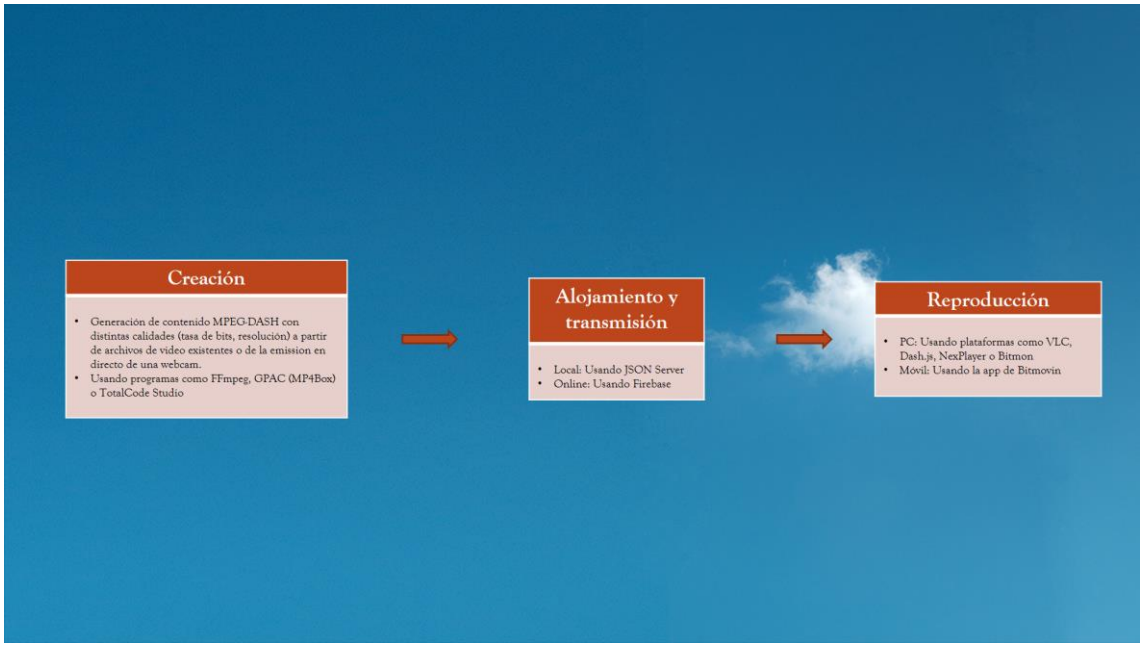

Figura 2: Esquema simplificado del proyecto

A continuación, siguiendo el esquema de la figura 2, se procederá a realizar un estudio de algunos de los principales programas utilizados para la creación de contenido DASH, que, basándose en el estándar, generan archivos con la codificación adecuada y necesaria para una transmisión cuya calidad se adaptará a la capacidad de la red, y también de varios programas capaces de reproducir este contenido DASH obtenido anteriormente.

Así mismo, será necesaria la utilización de herramientas adicionales para el establecimiento de vías de alojamiento local y online de este contenido para facilitar su transmisión.

Por último, para proveer todos los detalles de la investigación, se mostrarán las conclusiones obtenidas.

2. Qué es MPEG-DASH y cómo funciona

MPEG-DASH es una técnica eficiente de transporte para servicios de vídeo que utiliza distintas versiones del mismo contenido multimedia, tanto en vivo como bajo demanda, para adaptarse al ancho de banda disponible.

DASH es un estándar internacional (ISO/IEC 23009-1:2019) –y el primero– para streaming adaptativo desarrollado por MPEG en el que el cliente solicita información al servidor mediante peticiones HTTP, selecciona las herramientas necesarias para el streaming y monitoriza los cambios en la red para decidir cambios en la reproducción.
Las siglas de MPEG-DASH provienen de Moving Picture Experts Group - Dynamic Adaptive Streaming over HTTP.

MPEG es un grupo de trabajo de expertos que la ISO y la IEC formaron en 1988 para establecer estándares para la transmisión de audio y vídeo. Su metodología se considera asimétrica, ya que la codificación es mucho más compleja que la decodificación [8].

DASH es una técnica de streaming que divide el contenido en una secuencia de segmentos pequeños que se sirven sobre HTTP. Cada segmento contiene un intervalo corto del tiempo total de reproducción del contenido, que potencialmente puede durar horas, como una película o la retransmisión en directo de un evento deportivo. El contenido está disponible a una variedad de bitrates, es decir, segmentos alternativos codificados a diferentes bitrates que cubren intervalos cortos y alineados del tiempo total de la reproducción. Mientras el contenido es reproducido por un cliente, este cliente utiliza un algoritmo de adaptación de bitrate (ABR) para seleccionar automáticamente el segmento con el bitrate más grande posible que puede ser descargado a tiempo para su reproducción sin causar interrupciones [9].

Utiliza el protocolo de transporte TCP, así que se apoya en la estructura de servidores web HTTP ya existentes. Es compatible con HTML5. Esto significa que puede ejecutarse directamente en el navegador.

3. Elementos clave de la estructura MPEG-DASH

3.1. Segmento

El archivo que se quiere transmitir se divide en partes pequeñas, conocidas como segmentos, que se envían de forma constante, permitiendo la reproducción inmediata sin necesidad de recibir de antemano la totalidad del contenido. Un segmento contiene únicamente información de una parte del contenido, por ejemplo, vídeo, audio o texto [10].

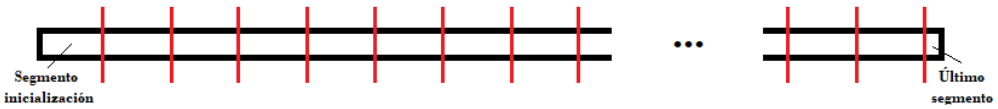

Figura 3: Segmento

El cliente reproduce los segmentos uno tras otro como si formasen un único archivo, aunque el segmento reproducido podrá tener una calidad diferente a la del anterior.

Lo habitual es crear segmentos de una duración determinada. Por tanto, el último segmento será el único con una extensión probablemente distinta a la de los demás.

Estos segmentos podrán ser archivos separados, pedazos físicos almacenados en el disco duro, en una emisión en vivo; o estar indicados en rangos de bytes dentro de un archivo único, común en una transmisión bajo demanda aunque no obligatorio. Cada segmento tiene un URI, una dirección para localizarlo en el servidor y descargarlo con un HTTP GET.

Los segmentos de inicialización contienen la configuración del codificador específica de esa serie de segmentos, información necesaria para decodificar los segmentos correspondientes, como su duración o el tipo de Stream Access Point. Cada segmento contiene al menos un Stream Access Point, que es un frame en el que la decodificación puede comenzar sin utilizar información de frames anteriores [11].

Los segmentos de una misma parte encadenados forman una Representation.

3.2. Representation

Una Representation es una codificación alternativa, una variante paralela prácticamente equivalente, de la misma parte del contenido, ya sea vídeo, audio, texto, etc. La guía de interoperabilidad dice explícitamente que las Representations han de ser no multiplexadas, es decir, cada representación solamente contendrá un único componente (vídeo o audio o texto, etc.) [12].

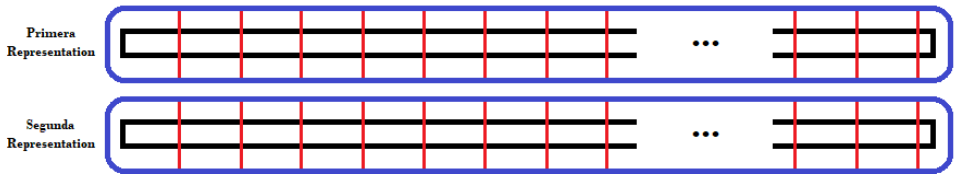

Figura 4: Representations.

Cada Representation puede diferir de las otras en características como la resolución o el bitrate de la imagen o el número de canales de audio, es decir, existe una Representation por cada versión del archivo original.
Una Representation puede constar de uno o más segmentos, que se enumerarán mediante un nombre único, un rango de bytes o una plantilla para poder acceder a ellos de forma lógica y rápida. Los segmentos deberán estar alineados en las distintas Representations, es decir, aunque tengan codificaciones diferentes, serán siempre los mismos para posibilitar las transiciones, tendrán la misma duración y comenzarán y finalizarán en el mismo punto.

Todas las Representations semejantes forman un Adaptation Set.

3.3. Adaptation Set

Un Adaptation Set agrupa una o varias Representations de la misma parte del contenido y provee información sobre ellas. Todas las Representations de vídeo forman el Adaptation Set de vídeo.

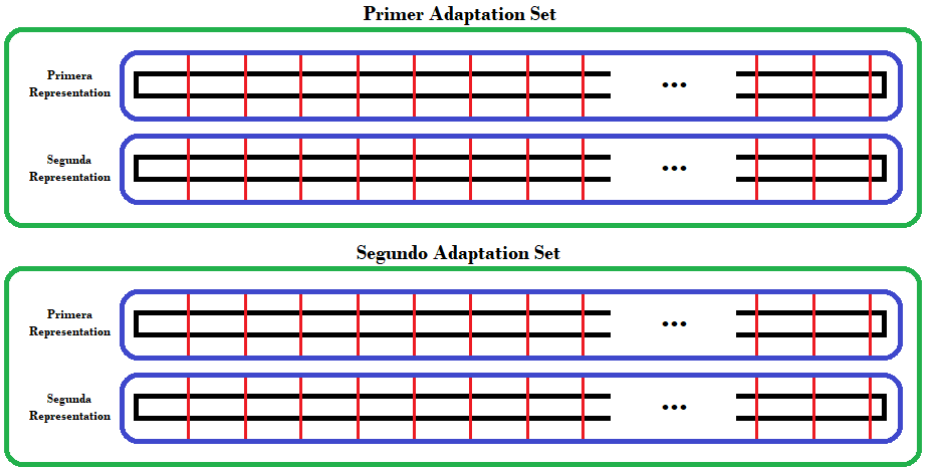

Figura 5: Adaptation Sets

Solo una Representation se reproducirá al mismo tiempo, dependiendo de las características de la conexión o de la configuración del Adaptation Set, que especifica el

tipo de contenido multimedia y si permite alternar la reproducción de sus Representations.

3.4. Period

Un Period contiene uno o varios Adaptation Sets, es decir, todas las partes que forman un contenido multimedia concreto (audio, vídeo, texto, etc.).

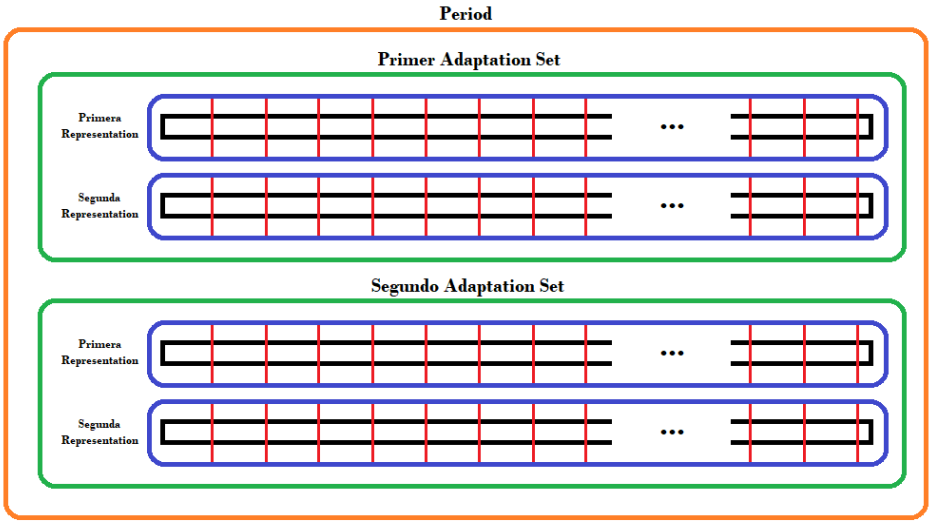

Figura 6: Period

Cada Period es un intervalo con un tiempo de inicio y una duración. Puede representar un capítulo, una escena o un anuncio intercalado de los que forman el total del contenido total de la transmisión, pero agrupa el vídeo, el audio y el resto de información necesaria para reproducir esa parte al completo. Pueden existir varios Periods, que se retransmitirían uno detrás de otro, y estarán indicados en el MPD [13].

3.5. MPD

El MPD o Manifiesto es un documento XML que contiene información sobre los Periods, Adaptation Sets, Representaciones y segmentos existentes para una transmisión del mismo contenido.

Es el cliente, que solicita el MPD directamente, quien selecciona el conjunto de Representations que utilizará basándose en su capacidad, la calidad de la conexión, elementos descriptivos del manifiesto y las preferencias del usuario.

El cliente construye una línea temporal, comienza a reproducir el contenido pidiendo los segmentos apropiados y, con el paso del tiempo y dependiendo de las características

antes mencionadas, es capaz de solicitar los segmentos posteriores, de la misma o de otra Representation, y unirlos permitiendo una reproducción sin interrupciones.

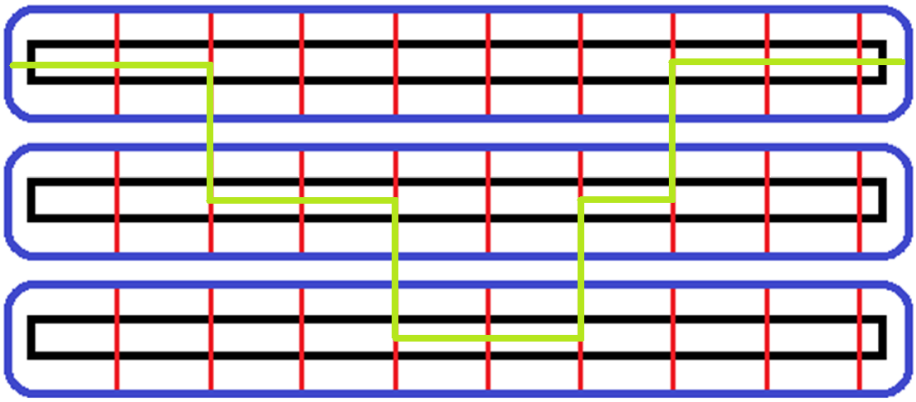

Figura 7: Reproducción sin interrupciones de segmentos de distintas Representations

En reproducciones en vivo, el MPD también provee el tiempo de disponibilidad de los segmentos.

3.5.1. Partes del manifiesto

3.5.1.1. MPD

XML es un lenguaje de marcado que define elementos para crear un lenguaje propio que pueda ser leído por distintas aplicaciones aunque sean incompatibles. Hace que el cliente entienda las marcas.

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:dash:schema:mpd:2011"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011http://standards.iso.org/ittf/Publ
     iclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
     profiles="urn:mpeg:dash:profile:isoff-live:2011"
     type="static"
     mediaPresentationDuration="PT1M56.8S"
     maxSegmentDuration="PT4.0S"
     minBufferTime="PT8.0S">
```

Figura 8: XML

En XML, los nombres de los elementos los define el desarrollador. Esto podría ocasionar problemas al mezclar XML de distintas aplicaciones. Los Namespaces proveen una forma de evitar estos conflictos.

Cuando se usan prefijos, se debe definir un namespace para el prefijo y una URI que le dará un nombre único al prefijo. Normalmente, la URI será una URL [14].

En este caso tenemos *xsi*, un esquema que define la descripción de la presentación de medios para MPEG-DASH explicando cada atributo.

El perfil sirve para restringir los formatos aplicados y habilitar la interoperabilidad y la señalización del uso de funciones. Puede ser entendido como un permiso para los clientes que implementan las funciones requeridas por el perfil para procesar la información (manifiesto y segmentos).

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:dash:schema:mpd:2011"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011http://standards.iso.org/ittf/Publ
     iclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
     profiles="urn:mpeg:dash:profile:isoff-live:2011"
     type="static"
     mediaPresentationDuration="PT1M56.8S"
     maxSegmentDuration="PT4.0S"
     minBufferTime="PT8.0S">
```

Figura 9: Perfil

La norma ISO/IEC 23009-1 permite a otras organizaciones o personas definir restricciones, permisos o extensiones usando este mecanismo. Estas definiciones externas se llaman puntos de interoperabilidad (IOP), y pueden señalarse en el perfil una vez se defina una URI, y el dueño de la URI es el responsable de proveer información suficiente en cuanto a las restricciones y los permisos de este punto de interoperabilidad [15].

En este caso, el perfil es el MPEG-DASH ISO Base media file format live profile, un perfil de transmisión en vivo, que no significa necesariamente que el contenido se esté generando en directo.

El perfil bajo demanda utiliza solamente un segmento por cada Representation, solo existe una pista de audio o vídeo por cada archivo. El perfil en vivo divide la Representation en segmentos. Esto es esencial para una conexión en directo pero también puede usarse en una transmisión bajo demanda, por ejemplo, para un reproductor que no soporte señalización con rangos de bytes. El codificador crea en el mismo directorio los segmentos y el manifiesto que los indexa mediante una plantilla [16].

Si se indica el elemento maxSegmentDuration, los segmentos no deberán tener una duración mayor de la indicada en ninguna Representation actual o futura bajo ninguna circunstancia. Si no se indica, la duración máxima será la mayor duración documentada en el MPD [17].

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:dash:schema:mpd:2011"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011http://standards.iso.org/ittf/Publ
     iclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
     profiles="urn:mpeg:dash:profile:isoff-live:2011"
     type="static"
     mediaPresentationDuration="PT1M56.8S"
     maxSegmentDuration="PT4.0S"
     minBufferTime="PT8.0S">
```

Figura 10: maxSegmentDuration

Es posible enumerar en el MPD cada segmento por su rango de bytes (SegmentBase) o por un nombre único (SegmentList). Esto se conoce como *indexed addressing*.

Si se utiliza *simple addressing*, donde se accede a los segmentos con URL creadas mediante una plantilla llamada SegmentTemplate y el MPD indica el tiempo medio de duración de un segmento, se permite una variación del 50 % en los puntos de inicio de los segmentos para compensar derivas. Si el empaquetador no puede compensar esa deriva, creará un Period nuevo y ajustará los parámetros del addressing para compensarla y señalará las Representations como period-connected o period-continuous (tendrán la misma id en Adaptation Set y Representation y serán técnicamente compatibles y tendrán segmentos de inicialización funcionalmente equivalentes). Podrá haber Representations nuevas no conectadas dentro de Periods conectados. Si un Period acaba o comienza en mitad de un segmento, este segmento estará señalizado en los dos Periods que lo contienen, pero solamente deberá reproducirse una vez.

Si se utiliza *explicit addressing*, la duración exacta de cada segmento aparece indicada con una plantilla llamada SegmentTimeline dentro de la SegmentTemplate.

Tanto en *explicit addressing* como en *simple addressing* es posible definir en el Period una secuencia infinita de referencias a segmentos que se extiende hasta el final del Period [18].

El elemento minBufferTime no indica al cliente cuánto tiempo ha de almacenar el contenido recibido, sino cuánto buffer debería tener en condiciones de red ideales. No describe el jitter de la red, sino el de la codificación de contenido.

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:dash:schema:mpd:2011"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011http://standards.iso.org/ittf/Publ
     iclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
     profiles="urn:mpeg:dash:profile:isoff-live:2011"
     type="static"
     mediaPresentationDuration="PT1M56.8S"
     maxSegmentDuration="PT4.0S"
     minBufferTime="PT8.0S">
```

Figura 11: minBufferTime

Si la Representation (empezando en cualquier segmento) se entrega a un bitrate constante equivalente al requerido, entonces cada tiempo de presentación estará disponible, como máximo, con un retraso del tiempo de buffer [18].

Sin otra información que nos guíe, el tiempo del buffer debería establecerse al mismo valor que el tamaño del segmento, que será igual o mayor que el tamaño del GOP. Puede establecerse a un valor menor que la duración del segmento más grande, pero no debe ser mayor. Esto ocasionaría parones en la reproducción.

En una implementación simple y directa, un cliente decide descargar el siguiente segmento basándose en el buffer disponible, la tasa de descarga estimada, el valor de buffer mínimo y el ancho de banda de la Representation. Es tarea del cliente elegir la Representation adecuada, y puede descargar una cuyo ancho de banda cumpla la fórmula

$$ancho\ de\ banda\ Representation\ \leq tasa\ de\ descarga\ estimada\ \frac{buffer\ disponible}{buffer\ mínimo}$$

3.5.1.2. Period

El primer Period comienza siempre en el tiempo 0, sin importar si el manifiesto es estático o dinámico.

```
<Period id="0" start="PT0.0S">
        <AdaptationSet id="0" contentType="video" startWithSAP="1"
         segmentAlignment="true" bitstreamSwitching="true" frameRate="25/1"
         maxWidth="1920" maxHeight="1080" par="16:9" lang="und">
                <Representation id="0" mimeType="video/mp4" codecs="avc1.640028"
                bandwidth="3000000" width="1920" height="1080" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="1" mimeType="video/mp4" codecs="avc1.64001e"
                bandwidth="900000" width="640" height="360" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="2" mimeType="video/mp4" codecs="avc1.64000b"
                bandwidth="100000" width="160" height="90" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
```

```
        </AdaptationSet>
```

Figura 12: start

Al comienzo de cada segmento se le conoce como Stream Access Point (SAP). Será siempre un frame I, que permite la unión de segmentos, aunque sean de distintas Representations, sin que se aprecie el corte [19]. Podrá haber más de un frame I en el segmento, es decir, el tamaño del GOP será igual o menor que el del segmento [20].

```
<Period id="0" start="PT0.0S">
        <AdaptationSet id="0" contentType="video" startWithSAP="1"
         segmentAlignment="true" bitstreamSwitching="true" frameRate="25/1"
         maxWidth="1920" maxHeight="1080" par="16:9" lang="und">
                <Representation id="0" mimeType="video/mp4" codecs="avc1.640028"
                bandwidth="3000000" width="1920" height="1080" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="1" mimeType="video/mp4" codecs="avc1.64001e"
                bandwidth="900000" width="640" height="360" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="2" mimeType="video/mp4" codecs="avc1.64000b"
                bandwidth="100000" width="160" height="90" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
        </AdaptationSet>
```

Figura 13: startWithSAP

En el elemento startWithSAP se indica un 1 o un 2 dependiendo de si el frame I lleva o no asociados frames RADL (Random Access Decodable Leading). Si el decodificador empieza a leer desde un frame que necesita frames previos para la predicción, puede corromperse, ya que el frame del segmento anterior es diferente si pertenece a otra Representation.

Un IDR es un frame especial de H.264 que especifica que ningún frame posterior a él puede hacer referencia a uno anterior al IDR. Así, la búsqueda del reproductor será más sensible y fácil. Cualquier IDR es un frame I, pero no viceversa. Puede haber frames I que no sean IDR. Si todos los frames I son IDR, entonces todos los GOP son cerrados. Si no, algunos GOP serán abiertos.

Por defecto, el códec H.264 produce GOP cerrados y H.265 los produce abiertos [21].

Los segmentos deben estar alineados. Se dice que están alineados si los puntos de inicio y fin de todos los segmentos son iguales en todas las Representations que pertenecen al mismo Adaptation Set.

```
<Period id="0" start="PT0.0S">
        <AdaptationSet id="0" contentType="video" startWithSAP="1"
         segmentAlignment="true" bitstreamSwitching="true" frameRate="25/1"
         maxWidth="1920" maxHeight="1080" par="16:9" lang="und">
                <Representation id="0" mimeType="video/mp4" codecs="avc1.640028"
                bandwidth="3000000" width="1920" height="1080" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="1" mimeType="video/mp4" codecs="avc1.64001e"
                bandwidth="900000" width="640" height="360" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="2" mimeType="video/mp4" codecs="avc1.64000b"
                bandwidth="100000" width="160" height="90" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
        </AdaptationSet>
```

Figura 14: segmentAlignment

Si se usa *simple addressing* o *explicit addressing*, debe aparecer el elemento segmentAlignment.

El bitstreamSwitching permite decodificar una secuencia unida de segmentos de distintas Representations en un mismo Adaptation Set sin que el decodificador tenga que reiniciarse o se entere siquiera del cambio.

```
<Period id="0" start="PT0.0S">
        <AdaptationSet id="0" contentType="video" startWithSAP="1"
         segmentAlignment="true" bitstreamSwitching="true" frameRate="25/1"
         maxWidth="1920" maxHeight="1080" par="16:9" lang="und">
                <Representation id="0" mimeType="video/mp4" codecs="avc1.640028"
                bandwidth="3000000" width="1920" height="1080" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
```

```
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation id="1" mimeType="video/mp4" codecs="avc1.64001e"
                bandwidth="900000" width="640" height="360" sar="1:1">
                    <SegmentTemplate timescale="12800" initialization="init-
                    stream$RepresentationID$.m4s" media="chunk-
                    stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                        <SegmentTimeline>
                            <S t="0" d="51200" r="28" />
                            <S d="10240" />
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
                <Representation id="2" mimeType="video/mp4" codecs="avc1.64000b"
                bandwidth="100000" width="160" height="90" sar="1:1">
                    <SegmentTemplate timescale="12800" initialization="init-
                    stream$RepresentationID$.m4s" media="chunk-
                    stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                        <SegmentTimeline>
                            <S t="0" d="51200" r="28" />
                            <S d="10240" />
                        </SegmentTimeline>
                    </SegmentTemplate>
                </Representation>
        </AdaptationSet>
```

Figura 15: bitstreamSwitching

Todas las Representations del mismo Adaptation Set de vídeo deberán ser codificaciones alternativas del mismo contenido original, codificadas de tal forma que el cambio entre ellas no se aprecie debido al tamaño de la imagen o la relación de aspecto.

La pixel aspect ratio (par), la relación de aspecto, deberá aparecer en el Adaptation Set.

La sample aspect ratio (sar), para escalar la imagen de modo que cada Representation se muestre exactamente igual, deberá aparecer en la Representation.

La anchura, la altura y la framerate deberán aparecer en uno de ellos pero no en ambos.

El ancho de banda es particular de cada Representation.

Las propiedades maxHeight y maxWidth no deberían aparecer y deberán ser ignoradas por el cliente, puesto que son datos triviales que pueden calcularse cuando sea necesario.

Si aparece la propiedad scanType, deberá indicar un escaneado progresivo, ya que el vídeo no progresivo no es interoperable.

La propiedad lang deberá aparecer en Adaptation Sets de audio o texto indicando el idioma, pero no es necesaria en Adaptation Sets de video [10].

3.5.1.3. Representation

El cliente tomará el elemento mimeType para identificar el método apropiado de renderización. Típicamente, el cliente seleccionará al menos un Adaptation Set de cada tipo. Además, debería usar el valor del elemento codecs para determinar si su plataforma de reproducción puede reproducir el contenido del Adaptation Set [10].

```
<Period id="0" start="PT0.0S">
        <AdaptationSet id="0" contentType="video" startWithSAP="1"
         segmentAlignment="true" bitstreamSwitching="true" frameRate="25/1"
         maxWidth="1920" maxHeight="1080" par="16:9" lang="und">
                <Representation id="0" mimeType="video/mp4" codecs="avc1.640028"
                bandwidth="3000000" width="1920" height="1080" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="1" mimeType="video/mp4" codecs="avc1.64001e"
                bandwidth="900000" width="640" height="360" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="2" mimeType="video/mp4" codecs="avc1.64000b"
                bandwidth="100000" width="160" height="90" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
        </AdaptationSet>
```

Figura 16: mimeType y codecs

- Un Adaptation Set de vídeo estará identificado por "video/mp4" y un codec admitido, por ejemplo: avc1 o avc3, de H.264; o hev1 o hvc1, de H.265.
- Un Adaptation Set de audio estará identificado por "audio/mp4" y un codec admitido, por ejemplo: mp4a.40.2, mp4a.40.5 o mp4a.40.29.
- Un Adaptation Set de texto estará identificado por "application/mp4" y un codec admitido o por "application/ttml+xml", que no contará con el parámetro codecs.
- Un Adaptation Set de metadatos interpretados por la aplicación estará identificado por "application/mp4" y un codec admitido.
- Un Adaptation Set de miniaturas estará identificado por "image/jpeg" o "imagen/png" y un schemeIdUri.

Timescale es el valor base de lectura de un flujo. Deberá estar presente en la SegmentTemplate (*simple addresing* o *explicit addressing*) o en la SegmentBase (*indexed addressing*).

```
<Period id="0" start="PT0.0S">
        <AdaptationSet id="0" contentType="video" startWithSAP="1"
         segmentAlignment="true" bitstreamSwitching="true" frameRate="25/1"
         maxWidth="1920" maxHeight="1080" par="16:9" lang="und">
                <Representation id="0" mimeType="video/mp4" codecs="avc1.640028"
                bandwidth="3000000" width="1920" height="1080" sar="1:1">
```

```
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="1" mimeType="video/mp4" codecs="avc1.64001e"
                bandwidth="900000" width="640" height="360" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="2" mimeType="video/mp4" codecs="avc1.64000b"
                bandwidth="100000" width="160" height="90" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
        </AdaptationSet>
```

Figura 17: timescale

Las unidades de escala de tiempo están definidas como número de unidades por segundo (hercios) [17]. Así, con una timescale de 12800 y una duración de segmento de 51200, sabemos que todos los segmentos, salvo el último, duran 4 segundos (51200/12800).

¿Por qué 12800? El valor por defecto 1 no es probable que coincida con cualquier contenido del mundo real y es mucho más probable que cause un error no intencionado. Valores pequeños pueden ocasionar problemas de sincronización de audio y video [22]. Los codificadores usan valores suficientemente grandes para evitar reinicios. Para elegir la escala de tiempo, se tiende a respetar el framerate para así evitar obviar frames o alterar el framerate de salida. Se suele multiplicar el framerate por 2 tantas veces como sea necesario para alcanzar un valor mayor de 10000 [23]. También es típico el uso de 90000, porque es divisible por 24, 25 y 30 (framerates habituales); y también de 48000, porque es una frecuencia típica de muestreo de audio [24]. En este caso, 12800 es divisible por 25, que es la framerate utilizada en el MPD que sirve como ejemplo.

Todas las Representations del mismo Adaptation Set deben tener la misma escala de tiempo. Deberá ser suficientemente pequeña para que, ni siquiera en transmisiones largas en directo, se exceda un valor de 2^{53}. Se espera que los codificadores usen una escala de tiempo apropiada y campos de timestamp suficientemente grandes para evitar reinicios. Si ocurre un reinicio, debe comenzarse un nuevo Period para poder establecer una nueva línea temporal [17].

El elemento inicialization contiene la plantilla de URL que hace referencia al segmento de inicialización de una Representation.

El elemento media contiene la plantilla de URL que hace referencia a los segmentos de la misma Representation.

```
<Period id="0" start="PT0.0S">
        <AdaptationSet id="0" contentType="video" startWithSAP="1"
         segmentAlignment="true" bitstreamSwitching="true" frameRate="25/1"
         maxWidth="1920" maxHeight="1080" par="16:9" lang="und">
                <Representation id="0" mimeType="video/mp4" codecs="avc1.640028"
                bandwidth="3000000" width="1920" height="1080" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="1" mimeType="video/mp4" codecs="avc1.64001e"
                bandwidth="900000" width="640" height="360" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="2" mimeType="video/mp4" codecs="avc1.64000b"
                bandwidth="100000" width="160" height="90" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
        </AdaptationSet>
```

Figura 18: initialization, media y startNumber

Si se usa la plantilla $Number$, se necesita el elemento startNumber para indicar el primer segmento (por defecto, 1). La numeración de segmentos forma una secuencia continua comenzando por el indicado en startNumber [25].

Si el reproductor necesita determinar el segmento más reciente, utilizará la fórmula

$$\frac{wall\ clock\ time - availabilityStartTime}{duration/timescale} + startNumber$$

Utilizando *explicit addressing* (con SegmentTimeline), se accede a los segmentos mediante URL construidas usando una plantilla definida en el MPD que indica la duración exacta de cada segmento, ya sea por tiempo de inicio o por número secuencial.

```
<Period id="0" start="PT0.0S">
        <AdaptationSet id="0" contentType="video" startWithSAP="1"
         segmentAlignment="true" bitstreamSwitching="true" frameRate="25/1"
         maxWidth="1920" maxHeight="1080" par="16:9" lang="und">
                <Representation id="0" mimeType="video/mp4" codecs="avc1.640028"
```

```
                bandwidth="3000000" width="1920" height="1080" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="1" mimeType="video/mp4" codecs="avc1.64001e"
                bandwidth="900000" width="640" height="360" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
                <Representation id="2" mimeType="video/mp4" codecs="avc1.64000b"
                bandwidth="100000" width="160" height="90" sar="1:1">
                        <SegmentTemplate timescale="12800" initialization="init-
                        stream$RepresentationID$.m4s" media="chunk-
                        stream$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                <SegmentTimeline>
                                        <S t="0" d="51200" r="28" />
                                        <S d="10240" />
                                </SegmentTimeline>
                        </SegmentTemplate>
                </Representation>
        </AdaptationSet>
```

Figura 19: segmentTimeline

Cada segmento puede tener una duración distinta incluso dentro de la misma Representation.

Cada elemento S es un segmento o una serie de segmentos con duración d. El tiempo de inicio t de un segmento se indica o se calcula con respecto al tiempo de inicio y la duración del anterior. La propiedad duration no debe aparecer en la SegmentTemplate. No debe haber ni huecos ni solapamientos entre segmentos.

El elemento r indica cuántos segmentos consecutivos, además del primero, tienen la misma duración. En el ejemplo, r=28 significa que hay 29 segmentos consecutivos de duración 51200. Los elementos r han de tener siempre un número positivo excepto el último indicado, que puede tener valor -1, que indicaría que las referencias continúan hasta un segmento que acaba en el punto de fin de Period o se solapa con él.

En las actualizaciones de un MPD dinámico pueden añadirse o quitarse elementos S, incrementarse el startNumber, añadirse el atributo t al primer elemento S o incrementarse el valor r del ultimo S, pero no se deberán modificar los S existentes [17].

El elemento audioSamplingRate indica la frecuencia de muestreo.

```
<AdaptationSet id="1" contentType="audio" startWithSAP="1" segmentAlignment="true"
bitstreamSwitching="true" lang="und">
        <Representation id="3" mimeType="audio/mp4" codecs="mp4a.40.2" bandwidth="128000"
         audioSamplingRate="44100">
                <AudioChannelConfiguration
                schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011"
                value="2" />
```

```
                <SegmentTemplate timescale="44100" initialization="init-
                stream$RepresentationID$.m4s" media="chunk-stream$RepresentationID$-
                $Number%05d$.m4s" startNumber="1">
                        <SegmentTimeline>
                                <S t="0" d="173056" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="1" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="36864" />
                        </SegmentTimeline>
                </SegmentTemplate>
        </Representation>
</AdaptationSet>
```

Figura 20: audioSamplingRate

Deberá indicarse en la Representation o en el Adaptation Set, pero no en ambos.

44100 Hz es un valor habitual de muestreo de audio, pero es posible encontrar varias.

En AudioChannelConfiguration se especifica la configuración del canal de audio.

```
<AdaptationSet id="1" contentType="audio" startWithSAP="1" segmentAlignment="true"
bitstreamSwitching="true" lang="und">
        <Representation id="3" mimeType="audio/mp4" codecs="mp4a.40.2" bandwidth="128000"
         audioSamplingRate="44100">
                <AudioChannelConfiguration
                schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011"
                value="2" />
                <SegmentTemplate timescale="44100" initialization="init-
                stream$RepresentationID$.m4s" media="chunk-stream$RepresentationID$-
                $Number%05d$.m4s" startNumber="1">
                        <SegmentTimeline>
                                <S t="0" d="173056" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="1" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="176128" r="2" />
                                <S d="177152" />
                                <S d="36864" />
                        </SegmentTimeline>
                </SegmentTemplate>
        </Representation>
</AdaptationSet>
```

Figura 21: AudioChannelConfiguration

Se espera que, al menos, los siguientes esquemas sean reconocidos por el cliente:

- urn:mpeg:dash:23003:3:audio_channel_configuration:2011
- urn:mpeg:dashB:cicp:ChannelConfiguration
- tag:dolby.com,2014:dash:audio_channel_configuration:2011

Los valores están definidos en tablas de configuración que especifican, por ejemplo, el mapeo de canales en relación a las posiciones de los altavoces [26].

3.5.2. Tipos de manifiesto

3.5.2.1. MPD estático

Un MPD estático ya está disponible en su totalidad desde su recepción, y la reproducción del contenido comenzará desde el inicio de la presentación. El manifiesto no depende de un mapeo en tiempo real.

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:dash:schema:mpd:2011"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011http://standards.iso.org/ittf/Publ
     iclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
     profiles="urn:mpeg:dash:profile:isoff-live:2011"
     type="static"
     mediaPresentationDuration="PT1M56.8S"
     maxSegmentDuration="PT4.0S"
     minBufferTime="PT8.0S">
```

Figura 22: MPD estático

El cliente puede reproducir cualquier parte de la presentación en cualquier momento, ya que el contenido total ya está disponible. Esto no significa que se hayan recibido todos los segmentos, sino que ya pueden ser solicitados. Las Representations deberán contener referencias a segmentos hasta cubrir la duración completa del contenido para posibilitar el envío del segmento necesario.

Los servicios bajo demanda utilizan un MPD estático, en el que es posible encontrar un único segmento por cada Representation [17].

El elemento mediaPresentationDuration deberá coincidir con la duración entre el punto 0 de la línea temporal y el final del último Period.

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:dash:schema:mpd:2011"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011http://standards.iso.org/ittf/Publ
     iclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
     profiles="urn:mpeg:dash:profile:isoff-live:2011"
```

```
      type="static"
      mediaPresentationDuration="PT1M56.8S"
      maxSegmentDuration="PT4.0S"
      minBufferTime="PT8.0S">
```

Figura 23: mediaPresentationDuration

De todas formas, los clientes no deberán confiar en este dato, deberán calcular la duración de un MPD estático sumando las duraciones de los Periods porque las duraciones de los XLinks (elementos XML externos al MPD que se cargan cuando es necesario) solo podrán conocerse tras cargarse. Por tanto, es imposible determinar siempre la duración total de un MPD en el lado del servidor, el cliente es el único que podrá conocerla [17].

3.5.2.2. MPD dinámico

En emisiones en directo se utiliza un MPD dinámico. La reproducción comenzará lo más cerca posible del directo, y el MPD podrá indicar que no permite al cliente retroceder para ver contenido emitido anteriormente.

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="urn:mpeg:dash:schema:mpd:2011"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011
      http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
      DASH_schema_files/DASH-MPD.xsd"
      profiles="urn:mpeg:dash:profile:isoff-live:2011"
      type="dynamic"
      minimumUpdatePeriod="PT4S"
      suggestedPresentationDelay="PT4S"
      availabilityStartTime="2021-01-22T10:34:38.954Z"
      publishTime="2021-01-22T10:34:43.005Z"
      maxSegmentDuration="PT4.0S"
      minBufferTime="PT8.0S">
      <ProgramInformation>
      </ProgramInformation>
```

Figura 24: MPD dinámico

Este MPD cambia con el tiempo -se van añadiendo los segmentos conforme se van creando- y, por tanto, el cliente debe solicitar la nueva versión con una regularidad indicada por el propio manifiesto. En cada actualización podrán añadirse, modificarse o eliminarse Periods. La línea temporal tiene un mapeo fijado al tiempo real, cada punto de su línea temporal corresponde a un punto del tiempo real.

Existe un periodo mínimo de actualización, valor llamado minimumUpdatePeriod, que especifica el tiempo más pequeño entre posibles cambios del MPD, así que debería coincidir con el tamaño de los segmentos para asegurar que el siguiente segmento ya está disponible. Un valor pequeño permite cambiar y anunciar nuevo contenido con menos antelación. Sin embargo, el cliente solicita actualizaciones con más frecuencia, aumentando el tráfico de subida y de bajada. Si su valor es 0, significa que el MPD no

sirve y el cliente deberá solicitar un nuevo manifiesto para poder adquirir nuevos segmentos. La ausencia de este dato indica una validez infinita del Period, que no tendrá más actualizaciones [27].

El cliente deberá calcular un retraso de presentación adecuado para asegurarse de que los segmentos ya están disponibles y de que hay tiempo suficiente para descargarlos. Cada cliente puede utilizar un algoritmo distinto para calcular este retraso. Si el MPD indica el suggestedPresentationDelay, que es opcional, el cliente deberá usarlo e ignorar el tiempo calculado. Así, se sincroniza aproximadamente la reproducción en todos los clientes. Debería tener un valor entre 2 y 4 veces la duración media de los segmentos, pero no menor de 4 segundos, para así mantener cierto buffer.

El punto 0 del MPD está asociado al punto del tiempo real indicado en availabilityStartTime, que es el punto de inicio de creación del manifiesto. Este valor no debe cambiar entre actualizaciones. Sí cambia el valor publishTime, que indica la hora a la que fue modificado el MPD. El primer Period debe empezar en el punto 0 de la línea temporal [17].

Para convertir un servicio live en onDemand, debería bastar con cambiar el elemento type a static, eliminar los elementos restringidos a un tipo dinámico y crear el elemento mediaPresentationDuration. No es necesario cambiar los segmentos, mantener sus URL puede ser beneficioso en términos de eficiencia [18].

4. Implementación de una arquitectura MPEG-DASH

4.1. Introducción

Los conocimientos adquiridos hasta ahora facilitarán el uso de las diferentes herramientas para el tratamiento de vídeo que es posible encontrar en Internet. Comprender el funcionamiento del estándar MPEG-DASH ayudará a seleccionar las opciones más necesarias y útiles en cada uno de ellos.

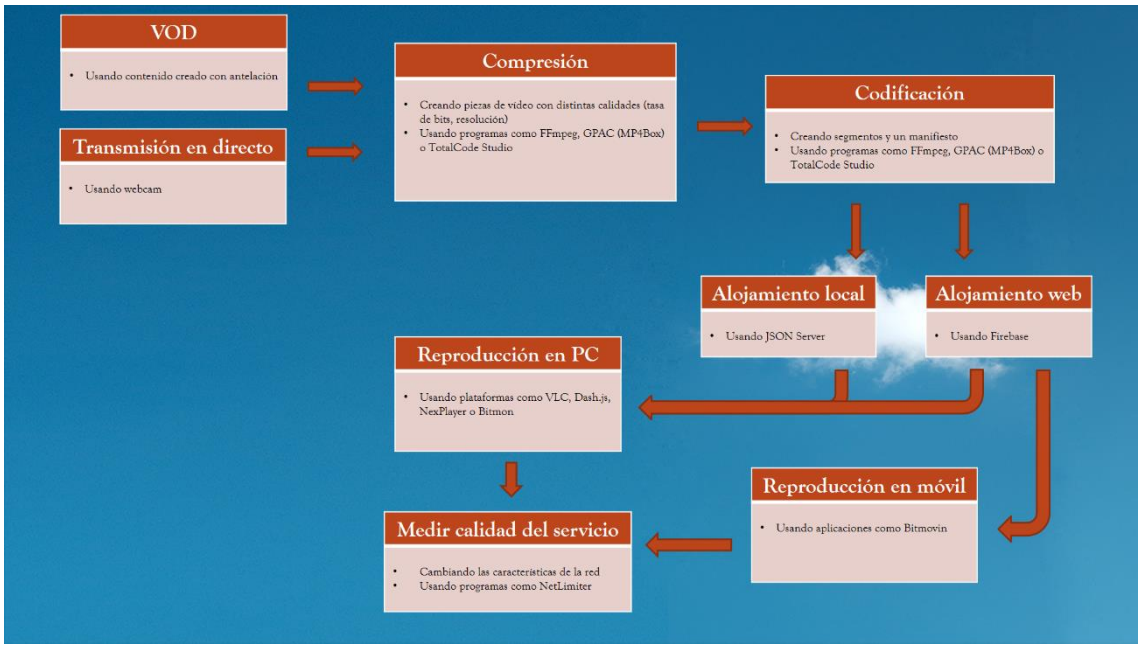

Figura 25: Esquema de trabajo del proyecto

Primeramente, se estudiarán varios programas de creación de archivos DASH. Con ellos, se crearán, a partir de distintas fuentes de información, las versiones alternativas que el cliente deberá seleccionar dependiendo del estado de su conexión. Además, codificarán adecuadamente estas versiones, es decir, generarán el manifiesto y los segmentos necesarios.

Para posibilitar la elaboración de una arquitectura completa de creación y reproducción de contenido DASH, será importante también aprender a manejar herramientas de hosting, ya que el contenido ha de alojarse en un servidor.

En tercera instancia, se analizará el uso de varias plataformas capaces de reproducir este contenido DASH generado.

Siguiendo el esquema de trabajo, el proceso de implementación se ha dividido en tres fases: la etapa de generación de vídeo, la de alojamiento y transmisión del contenido y la de reproducción del mismo. En cada una de ellas, se mostrarán las conclusiones obtenidas.

4.2. Generación de vídeo

El primer paso es crear contenido DASH que pueda reproducir un cliente. Se han analizado tres herramientas que sirven, entre muchas otras funciones, para generar versiones alternativas del mismo contenido, ya sea un archivo de vídeo o imágenes que procedan de la grabación en directo de una webcam, y el manifiesto y los segmentos que servirán para que el cliente pueda disponer de ellas.

En este proyecto, se van a crear tres versiones diferentes de cada archivo de vídeo tratado:

- La primera versión tendrá una resolución de 1920x1080 y bitrate de 3000 kbps
- La segunda versión tendrá una resolución de 640x360 y bitrate de 900 kbps
- La tercera versión tendrá una resolución de 160x90 y bitrate de 100 kbps

Se utilizará H.264, un códec de alta compresión muy popular creado en 2003 y desarrollado por MPEG [28].

Originalmente, la tercera versión iba a tener una resolución de 80x45, pero x264, uno de los programas utilizados, la rechaza porque no es compatible con el espacio de color i420 (YUV), el sistema de procesamiento de imagen. Se decidió utilizar las mismas tres versiones con cada herramienta.

4.2.1. TotalCode Studio

TotalCode Studio es un programa instalable que no necesita el uso de la línea de comandos. Es propiedad de la compañía MainConcept, nacida en 1993 [29].

4.2.1.1. Instalación

Es necesario descargar el programa aquí:

https://www.mainconcept.com/products/download-pages/for-professionals/totalcode.html

Basta con ejecutar el instalador y esperar a que se complete el proceso.

El programa tiene una interfaz con esta apariencia:

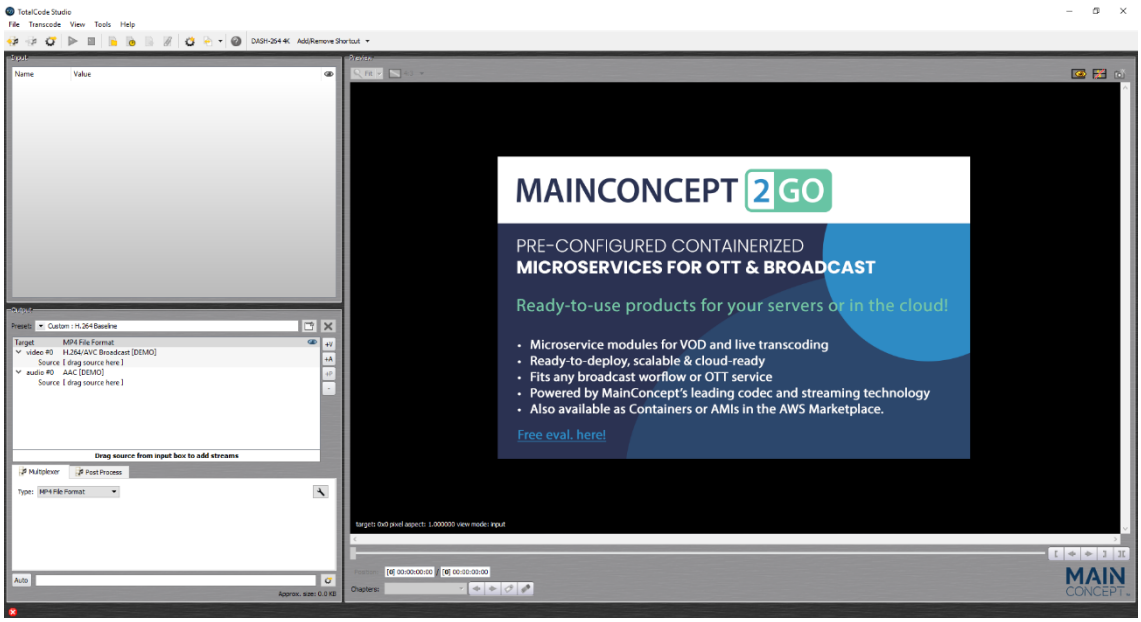

Figura 26: Interfaz de TotalCode Studio

4.2.1.2. Aplicación

4.2.1.2.1. Crear segmentos

Se debe elegir el vídeo de entrada, el formato de salida (DASH), las propiedades del vídeo de salida y la duración de los segmentos antes de comenzar la codificación.

DASH profile: Static Live (si se desea una versión segmentada) / VOD (si se desea un único segmento para una transmisión bajo demanda)

Segment duration: 4 sec

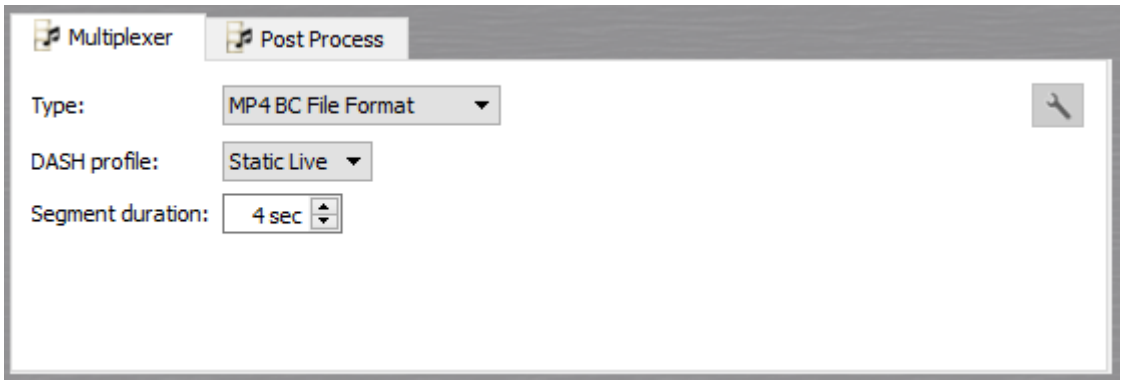

Figura 27: Multiplexer de TotalCode Studio

Preset: DASH: DASH-264 4K

Target: MP4 BC File Format

Framerate: 25p

Aspect: 16:9

	Anchura	**Altura**	**Target**	**Max**
Calidad 1	1920px	1080px	3000kbps	6000kbps
Calidad 2	640px	360px	900kbps	1800kbps
Calidad 3	160px	90px	100kbps	200kbps

I-frame interval: 25

Min. I-frame distance: 25

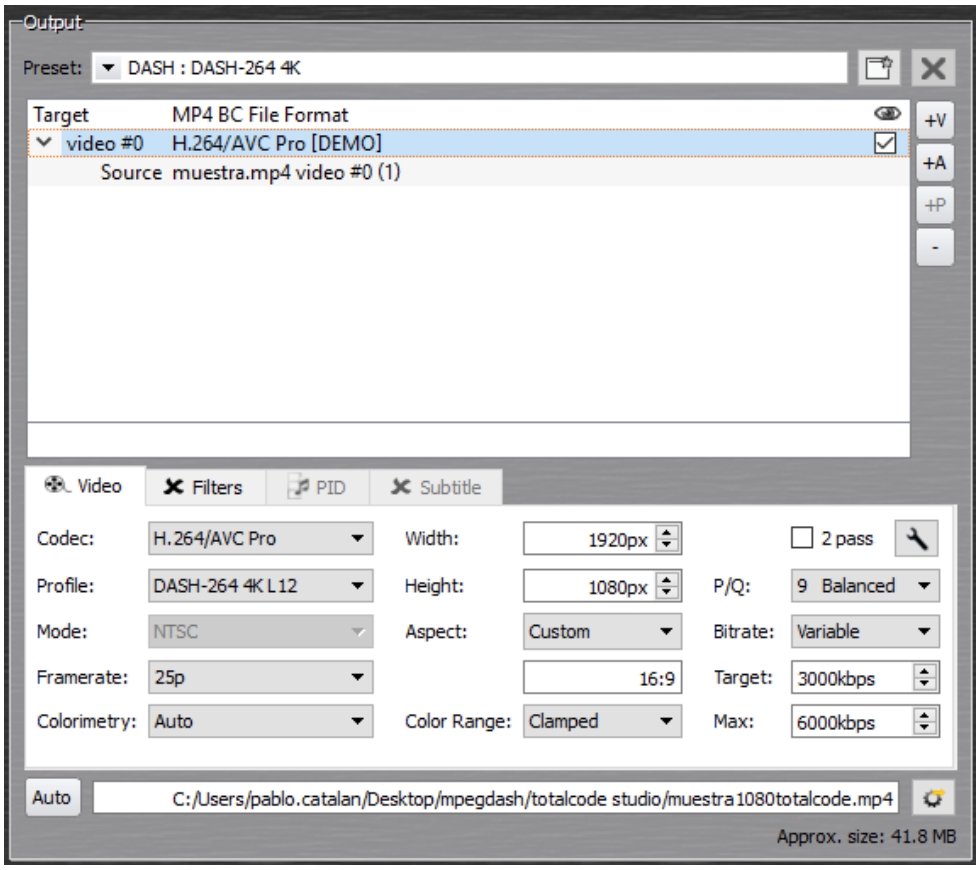

Figura 28: Output de TotalCode Studio

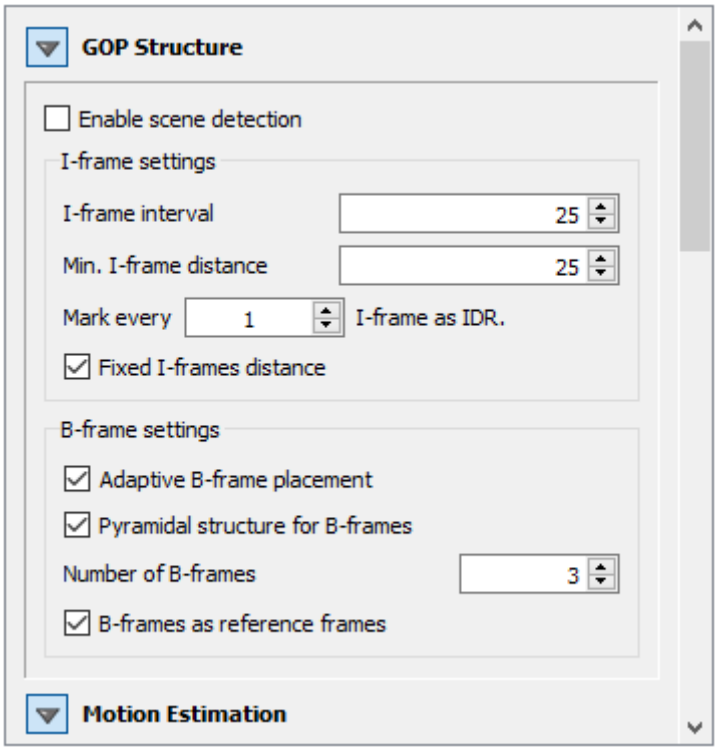

Figura 29: Propiedades avanzadas de TotalCode Studio

El programa genera, para cada versión, un manifiesto, con extensión .mpd; los segmentos correspondientes, con extensión .mp4; y los segmentos de inicialización, con extensión .mp4.

4.2.1.2.2. Crear MPD

Para combinar los manifiestos de cada versión, hay que acceder al apartado correspondiente.

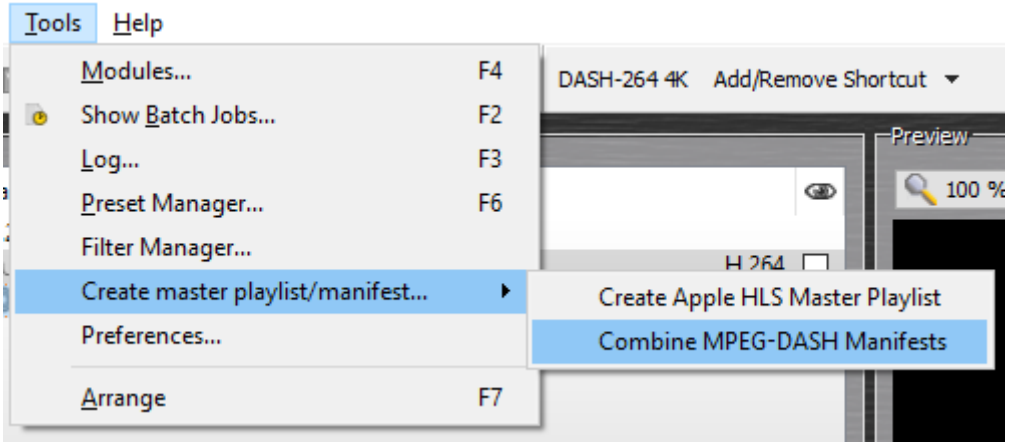

Figura 30: Crear manifiesto TotalCode Studio

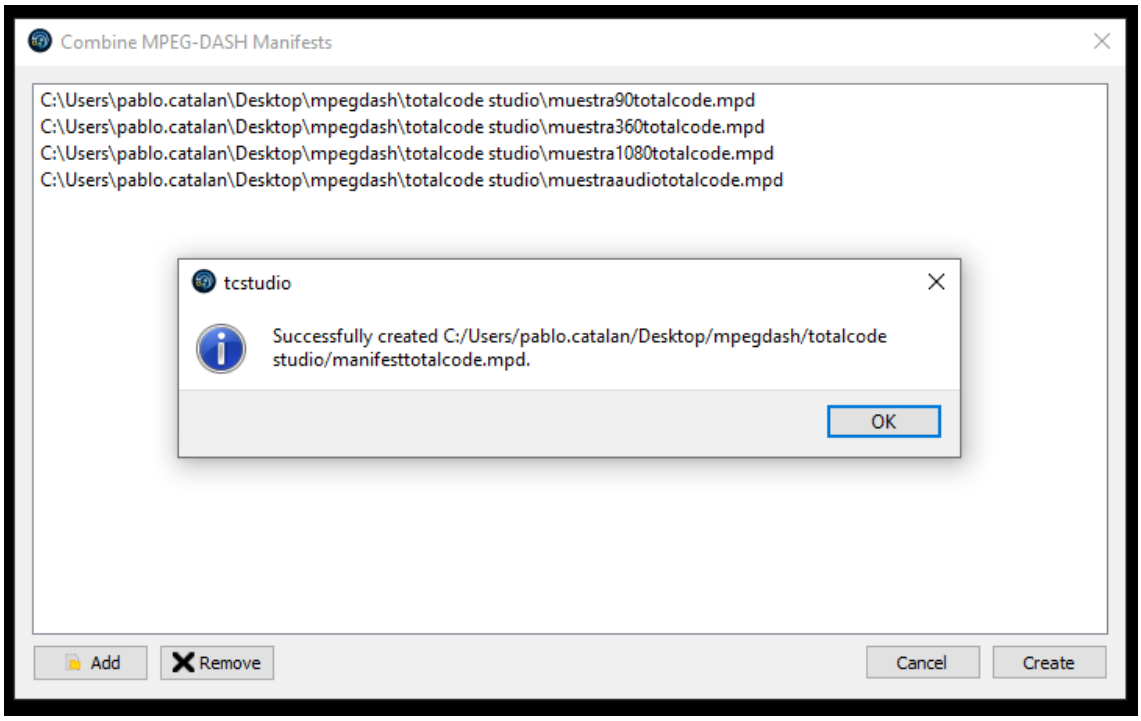

Figura 31: Manifiesto creado con TotalCode Studio

4.2.1.3. Conclusiones

La versión de prueba del programa incluye todos los formatos de la versión completa, pero añade una marca de agua en el vídeo y reduce las capacidades de audio, por ejemplo, suprime el sonido en intervalos alternos de 30 segundos.

El programa tiene herramientas avanzadas que también conviene utilizar, como la anulación de la detección de escenas o la selección del tamaño de GOP.

Es un programa fácil de usar con una interfaz bastante intuitiva. No está disponible en Mac.

4.2.2. MP4Box y x264

MP4Box es un conversor rápido y ligero de código abierto que se utiliza desde la línea de comandos. Está disponible desde 2004 y maneja contenido DASH desde 2011. Es propiedad de GPAC, una implementación del estándar de MPEG-4 escrita en ANSI C y fundada en 1999 que se convirtió en un proyecto de código abierto en 2003 [30].

X264 es una biblioteca libre para codificar flujos de vídeo H.264 y uno de los más avanzados para codificar H.264. Está desarrollada actualmente por VideoLAN [31].

4.2.2.1. Instalación

Es posible descargar el programa MP4Box aquí:

https://www.videohelp.com/software/MP4Box

Es posible descargar el archivo x264.exe aquí:

https://www.videolan.org/developers/x264.html

Basta con ejecutar el instalador de MP4Box y esperar a que se complete el proceso. A partir de ese momento, no será necesario acceder a una ubicación concreta para ejecutarlo, pero sí se necesitará indicar la ruta completa del archivo que se quiere codificar.

El archivo x264.exe no es necesario ejecutarlo, pero ha de colocarse en la misma carpeta que el archivo que se quiere codificar.

4.2.2.2. Aplicación

4.2.2.2.1. Recodificar vídeo

Primero se utiliza x264 para preparar el contenido de vídeo. Con los siguientes comandos, se recodifica en H.264 con las propiedades necesarias, como la resolución, el bitrate o la colocación de los frames I, para cada versión deseada [32].

Hay que tener cuidado con el tipo de comillas que se utilizan, ya que los símbolos “” no son reconocidos por el terminal del programa Visual Studio Code. Los símbolos correctos son "". En la consola de Windows sí que se interpretan correctamente ambos símbolos.

```
x264 --output intermediate1080.264 --fps 25 --preset slow --bitrate 3000 --vb
v-maxrate 6000 --vbv-bufsize 3000 --min-keyint 25 --keyint 25 --scenecut 0 --
no-scenecut --pass 1 --video-filter "resize:width=1920,height=1080" muestra.m
p4
```

```
x264 --output intermediate360.264 --fps 25 --preset slow --bitrate 900 --vbv-
maxrate 1800 --vbv-bufsize 900 --min-keyint 25 --keyint 25 --scenecut 0 --no-
scenecut --pass 1 --video-filter "resize:width=640,height=360" muestra.mp4
```

```
x264 --output intermediate90.264 --fps 25 --preset slow --bitrate 100 --vbv-m
axrate 200 --vbv-bufsize 100 --min-keyint 25 --keyint 25 --scenecut 0 --no-sc
enecut --pass 1 --video-filter "resize:width=160,height=90" muestra.mp4
```

Opciones utilizadas:

`--output intermediate1080.264`: especifica el nombre del archivo de salida, en este caso intermediate1080. La extensión de este archivo es .264 porque es un flujo de H.264/AVC sin procesar.

`--fps 25`: especifica los frames por segundo del archivo de salida, en este caso 25. Si se cambia el framerate con respecto al archivo de entrada, MP4Box alterará la duración y la velocidad del vídeo causando una desincronización con el audio.

`--preset slow`: indica al programa la velocidad a la que debería intentar mejorar la calidad durante la compresión. Es una buena opción utilizar la opción lenta.

`--bitrate 3000`: indica la tasa de bits en kbps que debería alcanzarse, en este caso 3000.

`--vbv-maxrate 6000`: especifica la tasa máxima de bits que deberá utilizarse en la codificación, en este caso 6000.

`--vbv-bufsize 3000`: indica el tamaño del búfer, en este caso 3000.

`--keyint 25`: indica el intervalo máximo entre fotogramas clave, en este caso 25. Es importante porque luego, al dividir el vídeo, se necesitará un frame I al inicio de cada segmento. El número resultante de multiplicar el tamaño del segmento por el framerate ha de ser múltiplo del tamaño del GOP, pero este valor no ha de cambiar necesariamente si se decide cambiar la duración de los segmentos.

`--min-keyint 25`: indica el intervalo mínimo entre fotogramas clave, en este caso 25. Con esta y la instrucción anterior, se fuerza la colocación de un fotograma clave en el punto exacto deseado. Es igualmente importante.

`--scenecut 0 --no-scenecut`: desactiva la decisión de keyframe adaptativo. Los fotogramas clave se colocarán donde se ha indicado, sin importar la información de la escena. Así se obtienen segmentos de la misma duración.

`--pass 1`: indica el de codificación que ha de utilizarse. Con 1, se usa la misma cantidad de datos para codificar una escena en negro y otra con muchos detalles. Con 2, la tasa de bits se adapta al contenido de la escena.

`--video-filter "resize:width=1920,height=1080"`: cambia la resolución del vídeo, en este caso 1920x1080. Si no es necesario, se puede omitir.

4.2.2.2.2. Introducir el flujo en un contenedor mp4

Con los siguientes comandos, se obtienen tres archivos independientes de vídeo, cada uno con sus propiedades.

```
mp4box -add intermediate1080.264 -fps 25 muestra1080.mp4
```

```
mp4box -add intermediate360.264 -fps 25 muestra360.mp4
```

```
mp4box -add intermediate90.264 -fps 25 muestra90.mp4
```

Opciones utilizadas:

`-add intermediate1080.264`: indica el flujo H.264 que queremos introducir en el contenedor mp4.

`-fps 25`: indica la framerate. Es recomendable especificarla porque H.264 no provee metadatos sobre ella. La cifra debe coincidir con la utilizada en el comando anterior.

`muestra1080.mp4`: especifica el nombre del archivo mp4 de salida.

4.2.2.2.3. Crear segmentos y MPD

Con el siguiente comando, se generan un manifiesto, con extensión .mpd; los segmentos correspondientes a cada resolución, con extensión .m4s; y los segmentos de inicialización, con extensión .mp4 [33].

```
mp4box -dash 4000 -rap -dash-profile dashavc264:live -bs-switching no -url-te
mplate -segment-timeline -out manifestmp4box.mpd -frag 4000 muestra.mp4#audio
muestra1080.mp4 muestra360.mp4 muestra90.mp4
```

Opciones utilizadas:

`-dash 4000`: indica la duración en milisegundos de los segmentos en los que se dividirá el archivo original, en este caso 4000.

`-frag 4000`: especifica la duración de los subsegmentos en los que se dividirán los segmentos. Si no se indica, se tomará el mismo dato que en la opción `dash`, es decir, creará un fragmento por cada segmento. Los fragmentos son una propiedad de archivos

MP4 fragmentados. Un MP4 fragmentado es más seguro porque los fragmentos pueden decodificarse por separado. Un caso de buen uso es una grabación de larga duración, porque en caso de fallo eléctrico, un MP4 fragmentado incompleto todavía es útil. Un segmento y un fragmento pueden parecer similares, pero los segmentos son archivos físicos y los fragmentos son pedazos lógicos de datos en un segmento, así que es posible tener varios fragmentos dentro de un segmento. En MPEG-DASH es preferible alinear los fragmentos con los segmentos [34].

`-rap`: fuerza a cada segmento a comenzar en un fotograma clave. La duración de los segmentos no cambiará porque previamente se ha forzado la colocación exacta de estos fotogramas.

`-dash-profile dashavc264:live`: crea un perfil en vivo (no necesariamente para transmisión en vivo). Puede utilizarse la opción onDemand.

`-bs-switching no`: indica que ha de crearse un segmento de inicialización por cada calidad. La otra opción no es `yes`, sino una serie de opciones para establecer el modo de unión de los flujos con distintas opciones, como la creación de un único segmento de inicialización [35].

`-url-template`: indica el uso de la plantilla de segmentos SegmentTemplate en lugar de especificarlos uno a uno. Se ignora esta opción si los segmentos se almacenan en el fichero de salida.

`-segment-timeline`: indica el uso de SegmentTimeline al enumerar los segmentos para especificar sus duraciones exactas.

`-out manifestmp4box.mpd`: especifica el nombre del fichero del MPD de salida.

Otras opciones útiles:

`-segment-name: segment-muestra-$RepresentationID$-$Number$$Init=inicio$`: especifica el nombre de los segmentos. Sustituirá la cadena $RepresentationID$ por el número de Representation y la cadena $Number$ por el número de segmento y añadirá el nombre elegido a los segmentos de inicialización. También pueden utilizarse las cadenas $Bandwidth$ y $Time$, que indicarán el ancho de banda de la Representation y el tiempo de comienzo del segmento.

`-mpd-title manifiesto`: especifica el nombre que se le dará a la propiedad Title del manifiesto, en este caso "manifiesto". Se puede omitir.

`-min-buffer 2000`: especifica el buffer mínimo en milisegundos, en este caso 2000.

`-dts muestra.mp4`: crea un fichero de texto que contiene información sobre cada frame del archivo indicado. Por ejemplo, RAP=1 indica que es un frame I. Así se puede saber la duración del GOP. No lee ni segmentos DASH ni flujos .264.

4.2.2.3. Conclusiones

Las versiones anteriores a la 0.9.0 de MP4Box no recodifican vídeo y audio, y es necesario utilizar herramientas externas. Hasta entonces, MP4Box no podía cambiar la resolución del vídeo directamente. Sí podía cambiar la relación de aspecto de píxel, aunque no todos los reproductores tenían por qué reproducir el archivo correctamente. Además, la imagen sufría deformaciones. Es por eso por lo que primero se utilizaba x264 para transformar el vídeo [36]. A partir de la versión 0.9.0 puede utilizarse MP4Box en combinación con filtros de codificación [37].

En sus nuevas versiones, MP4Box no necesitaría x264 para recodificar vídeo [38], pero las pruebas realizadas no han arrojado los resultados deseados y se han dado los mismos problemas indicados por otros usuarios [39].

Por el momento, MP4Box no puede, por sí solo, convertir a formato DASH una transmisión en directo [40]. Una opción sería tomar segmentos ya existentes y convertirlos a DASH en tiempo real [41].

Para el resto de funciones, es una herramienta de uso fácil y claro.

4.2.3. FFmpeg

FFmpeg es un software libre y de código abierto desarrollado en GNU/Linux desde el año 2000 pero compatible con Windows. Incluye la librería libavcodec, utilizada por muchos otros programas. Soporta muchos filtros y codecs y es compatible con los formatos de audio y vídeo más utilizados [42]. Convierte y codifica incluso transmisiones en directo. Programas como Audacity, OpenShot o Shortcut usan FFmpeg. Funciona con línea de comandos.

4.2.2.1. Instalación

Es posible descargar el archivo aquí:

https://ffmpeg.org/download.html

Es necesario extraer el contenido del archivo comprimido.

Para ejecutar el programa, desde la línea de comandos, se necesita acceder a la ubicación /bin, donde se encontrarán los archivos ffmpeg, ffplay y ffprobe, y colocar también allí los archivos que se quieran codificar.

4.2.2.2. Aplicación

4.2.2.2.1. Crear segmentos y MPD

Basándose en la extensión del archivo que ha de tratar, FFmpeg deduce el códec que debe emplear y usa los parámetros predeterminados, como la tasa de bits o el número de frames por segundo.

En FFmpeg, las opciones utilizadas afectan al archivo al que preceden, es decir, todas los que se refieran al archivo de entrada han de incluirse antes de indicar el archivo de entrada.

Con el siguiente comando, se generan un manifiesto, con extensión .mpd; y los segmentos correspondientes a cada versión y sus segmentos de inicialización, todos con extensión .m4s.

```
ffmpeg -y -re -i muestra.mp4 -keyint_min 25 -g 25 -sc_threshold 0 -r 25 -c:a a
ac -c:v libx264 -pix_fmt yuv420p -map v:0 -s:0 1920x1080 -b:v:0 3000k -maxrat
e:0 6000k -bufsize:0 3000k -map v:0 -s:1 640x360 -b:v:1 900k -maxrate:1 1800k
-bufsize 900k -map v:0 -s:2 160x90 -b:v:2 100k -maxrate:2 200k -bufsize:2 100
k -map 0:a -use_template 1 -use_timeline 1 -seg_duration 4 -frag_duration 4 -
adaptation_sets "id=0,streams=v id=1,streams=a" -f dash manifestffmpeg.mpd
```

Opciones utilizadas:

`-y`: pide sobrescribir los archivos de salida sin preguntar [43]. Durante la elaboración de este proyecto han surgido problemas que se han solucionado al incluir esta opción. Evita alertas indeseadas del programa relacionadas con segmentos multiplexados.

`-re`: pide que se lea y procese el archivo de entrada en tiempo real, en lugar de lo más rápido posible, simulando un flujo de entrada en directo. No debería utilizarse en transmisiones en directo porque podría ocasionar pérdida de paquetes [44].

`-i muestra.mp4`: indica el archivo de entrada, en este caso muestra.mp4.

`-keyint_min 25`: especifica el tamaño mínimo del GOP, la distancia mínima entre frames I.

`-g 25`: especifica el tamaño máximo del GOP, la distancia máxima entre frames I, en este caso 25. Con la combinación de las dos instrucciones, se consigue marcar exactamente la posición de estos frames. El reproductor buscará un frame I para cambiar de Representation, así que han de estar alineados en todas ellas. El número resultante de multiplicar el tamaño del segmento por el framerate ha de ser múltiplo del tamaño del GOP, pero este valor no ha de cambiar necesariamente si se decide cambiar la duración de los segmentos [45][46][47].

`-sc_threshold 0`: anula la sensibilidad de la detección de escenas de x264. Así, se consiguen bloques de igual tamaño, algo importante para el buffering.

`-r 25`: especifica los frames por segundo de la salida. Si cambia el framerate con respecto al archivo de entrada, ffmpeg duplica o elimina algunos frames para no alterar la duración y la velocidad del vídeo [44]. Esto puede ocasionar que movimientos suaves en el vídeo original se vean torpes o poco naturales. Es posible alterar la duración del vídeo de salida, pero no procede.

`-c:a aac`: indica el codificador de audio, que será aac [48].

`-c:v libx264`: indica el codificador de vídeo, que será H.264.

`-pix_fmt yuv420p`: establece un formato de píxel en el que los valores YUV están agrupados juntos en lugar de intercalados. Así, la imagen se puede comprimir más [49]. Se utiliza esencialmente para evitar un formato de píxel que no leerán los navegadores en caso de que el archivo de entrada sea .mov [50].

`-map v:0`: se usa para controlar manualmente la selección del flujo en cada archivo de salida. Cuando se usa, solamente se incluyen en ese archivo de salida los flujos seleccionados por el usuario [44]. Durante la elaboración de este proyecto se ha comprobado que es más seguro y efectivo utilizar la opción `map` delante de cada versión del contenido que vaya a crearse.

`-s:0 1920x1080`: especifica la resolución del primer flujo de vídeo, en este caso 1920x1080.

`-b:v:0 3000k`: indica el bitrate promedio deseado del primer flujo de vídeo, permitiendo controlar el ancho de banda utilizado, en este caso 3000 kbps.

`-maxrate:0 6000k`: indica el bitrate máximo del primer flujo de vídeo.

`-bufsize:0 3000k`: indica el búfer máximo del decodificador, que determina la variabilidad del bitrate de salida, en este caso 3000 kbps. Basándose en esta opción, FFmpeg calculará y corregirá el bitrate promedio producido. Si no se especificase, los intervalos serían mayores y habría saltos de bitrate por encima y por debajo del promedio. Con un búfer pequeño, se comprobará con más frecuencia el bitrate de salida y se limitará al promedio especificado, reduciendo la variación producida. Si es demasiado pequeño, podría ocasionar baja calidad de imagen, conformándose con las limitaciones y no pudiendo hacer optimizaciones, por ejemplo, basadas en repeticiones de frames. Se sugiere empezar por el mismo valor de bitrate promedio, o incluso la mitad, e ir aumentándolo hasta que los saltos de bitrate sean significativos. Restringirá el bitrate para no exceder cierto umbral que necesitaría mayor tiempo de transmisión y haría que el buffer del decodificador se desbordase esperando la llegada de nuevos datos [51].

`-map 0:a`: selecciona el flujo de audio, porque en nuestra implementación solo se requiere un mismo archivo de audio para todos los tamaños de vídeo.

`-use_template 1`: habilita el uso de SegmentTemplate en lugar de SegmentList. Así no se enumerará cada segmento en el manifiesto, sino que se usará una plantilla.

`-use_timeline 1`: habilita el uso de SegmentTimeline dentro de las SegmentTemplate. Permite especificar duraciones exactas de cada segmento en la plantilla.

`-seg_duration 4`: indica la duración en segundos de los segmentos, en este caso 4.

`-frag_duration 4`: indica la duración en segundos de los fragmentos dentro de los segmentos. Los fragmentos son una propiedad de archivos MP4 fragmentados. Un MP4 fragmentado es más seguro porque los fragmentos pueden decodificarse por separado. Un caso de buen uso es una grabación de larga duración, porque en caso de fallo eléctrico, un MP4 fragmentado incompleto todavía es útil. Un segmento y un fragmento pueden parecer similares, pero los segmentos son archivos físicos y los fragmentos son pedazos lógicos de datos en un segmento, así que es posible tener varios fragmentos dentro de un segmento. En DASH es preferible alinear los fragmentos con los segmentos [34].

`-adaptation_sets "id=0, streams=v id=1,streams=a"`: asigna los flujos correspondientes a cada Adaptation Set.

`-f dash manifestffmpeg.mpd`: indica que se utilizará el formato dash. Se indica también el nombre del archivo MPD de salida.

Otras opciones útiles:

`-n`: sirve para no sobrescribir archivos de salida y salir inmediatamente, pero existe un bug que provoca que la opción no funcione correctamente cuando se usa el tee muxer [52]. El tee muxer es la característica que usa ffmpeg para escribir los mismos datos en varias salidas, como por ejemplo al transmitir un vídeo por la red y guardar archivos en el disco duro al mismo tiempo [53]. No es lo mismo que especificar varias salidas en la línea de comandos. Con el tee muxer, los datos de audio y video se codificarán solo una vez. Con múltiples salidas convencionales, se inician múltiples operaciones de codificación, un proceso mucho más costoso. Como el tee muxer no representa un formato de salida particular, ffmpeg no puede autoseleccionar flujos de salida, así que todos los flujos que pretendan crearse deberán especificarse con la opción `map` [54].

`-profile:v:0 baseline`: otorga al flujo el perfil baseline de H.264 (subconjunto de características, como reproducción de color o compresión de vídeo adicional) al primer flujo de vídeo. Baseline es considerado típicamente el menos eficiente pero también el menos exigente. Tiene una ratio de compresión baja y soporta frames I y P. Otras opciones podrían ser `main` y `high`. El perfil main es considerado típicamente más eficiente en cuanto a ancho de banda pero también más exigente, y soporta frames I, P y B. El perfil high es el más complejo, tiene una ratio de compresión alta, soporta frames I, P y B y suele usarse para Blu-ray y HD-DVD [55][56].

`-bf 1`: limita el número máximo de frames B (entre 0 y 16).

`-b_strategy 0`: indica el nivel de decisión de uso de frames B. La opción `0` es muy rápida pero no es especialmente recomendable y no funciona si se utiliza la detección de escenas. La opción `1` es un buen balance entre velocidad y calidad. La opción `2` es más lenta pero más exacta, y se recomienda no usar una cantidad alta de frames B [47].

`-ar:a:1 22050`: especifica una frecuencia de muestreo de audio, en este caso de 22050 Hz.

`-vstats_file`: sirve para crear un fichero de texto con información útil sobre los frames. Si, pese a la utilización de esta opción, no se ha creado el fichero, se puede recurrir al comando de MP4Box indicado anteriormente, que nos dará la misma información.

`-init_seg_name init-muestra-$RepresentationID$.m4s`: indica el nombre del segmento de inicialización. Sustituirá $RepresentationID$ por el número de la Representation.

`-media_seg_name segment-muestra-$RepresentationID$-$Number$.m4s`: indica el nombre de los segmentos. Sustituirá $RepresentationID$ por el número de Representation y $Number$ por el número de segmento. También pueden utilizarse $Bandwidth$ y $Time$, que indicarán el ancho de banda de la Representation y el tiempo de comienzo del segmento.

4.2.2.2.2. Crear segmentos y MPD para emisión en directo

Con el siguiente comando, se obtiene una lista de dispositivos de vídeo y audio.

```
ffmpeg -list_devices true -f dshow -i dummy
```

En este caso aparecen como disponibles una webcam "Logitech HD Webcam C270" y un micrófono integrado en la webcam llamado "Micrófono (2- HD Webcam C270)". Esos son los nombre que deberán usarse en el próximo comando.

Opciones utilizadas:

-list devices true -f dshow: sirve para listar todos los dispositivos de DirectShow, un framework multimedia de Microsoft que puede reproducir y grabar archivos de vídeo bajo demanda. Acepta como entrada dispositivos de audio o vídeo, dispositivos de captura de vídeo o dispositivos analógicos de sintonización de televisión [57].

-i dummy: fuerza salir al acabar. No muestra los resultados sin esta opción.

Con el siguiente comando, se comienza un proceso de generación en directo de un manifiesto, con extensión .mpd; y los segmentos correspondientes a cada versión y sus segmentos de inicialización, todos con extensión .m4s.

```
ffmpeg -y -f dshow -video_size 1280x720 -framerate 25 -i video="Logitech HD W
ebcam C270":audio="Micrófono (2- HD Webcam C270)" -vcodec libx264 -r 25 -keyi
nt_min 0 -g 25 -map  v:0 -s:0 1280x720 -b:v:0 3000k -maxrate:0 6000k -bufsize
:0 3000k -map v:0 -s:1 640x360 -b:v:1 900k -maxrate:1 1800k -bufsize:1 900k -
map v:0 -s:2 160x90 -b:v:2 100k -maxrate:2 200k -bufsize:2 100k -map 0:a -use
_template 1 -use_timeline 1 -seg_duration 4 -frag_duration 4 -adaptation_sets
"id=0,streams=v id=1,streams=a" -f dash manifestlive.mpd
```

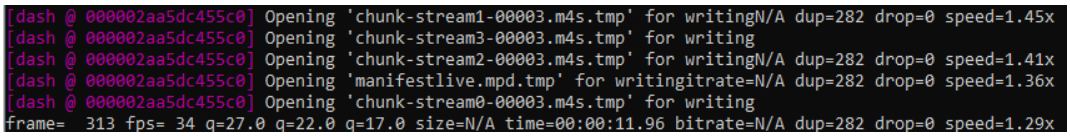

Figura 32: Creación de segmentos en directo

Opciones utilizadas:

`-y`: pide sobrescribir los archivos de salida sin preguntar [43]. Durante la elaboración de este proyecto han surgido problemas que se han solucionado al incluir esta opción. Evita alertas indeseadas del programa relacionadas con segmentos multiplexados.

`-f dshow` sirve para tomar como entrada dispositivos de DirectShow, un framework multimedia de Microsoft que puede reproducir y grabar archivos de vídeo bajo demanda. Acepta como entrada dispositivos de audio o vídeo, dispositivos de captura de vídeo o dispositivos analógicos de sintonización de televisión [57].

`-video_size 1280x720`: indica la resolución del flujo de vídeo de entrada. La webcam utilizada en este proyecto no soporta una resolución mayor ni cualquiera elegida al azar. Sí permite una resolución de 640x360.

`-framerate 25`: fuerza el framerate de entrada. Obliga al dispositivo a entregar imágenes con esta tasa. La webcam utilizada en este proyecto no soporta una tasa mayor de 30 frames/segundo.

`-i video="Logitech HD Webcam C270":audio="Micrófono (2- HD Webcam C270)"`: especifica como flujos de entrada el vídeo y el audio de la webcam disponible, en este caso los conectados al PC durante la elaboración del proyecto.

`-vcodec libx264`: indica el codificador de vídeo, que será H.264.

`-r 25`: especifica los frames por segundo de la salida. Es importante no alterar el framerate que se le ha exigido al dispositivo de entrada porque provoca un

desbordamiento continuo del buffer a tiempo real de la cámara, la pérdida de frames y, en consecuencia, saltos en la imagen.

`-keyint_min 0`: no se exige un tamaño mínimo de GOP.

`-g 25`: exige un tamaño máximo de GOP, la distancia máxima entre frames I, en este caso 25. Aunque varíe el tamaño del GOP a elección del codificador, nunca será mayor de 25 frames.

`-map v:0`: se usa para controlar manualmente la selección del flujo en cada archivo de salida. Cuando se usa, solamente se incluyen en ese archivo de salida los flujos seleccionados por el usuario [44]. Durante la elaboración de este proyecto se ha comprobado que es más seguro y efectivo utilizar la opción `map` delante de cada versión del contenido que vaya a crearse.

`-s:0 1280x720`: especifica la resolución del primer flujo de vídeo, en este caso 1280x720.

`-b:v:0 3000k`: indica el bitrate promedio deseado del primer flujo de vídeo, permitiendo controlar el ancho de banda utilizado, en este caso 3000 kbps.

`-maxrate:0 6000k`: indica el bitrate máximo del primer flujo de vídeo.

`-bufsize:0 3000k`: indica el búfer máximo del decodificador, que determina la variabilidad del bitrate de salida, en este caso 3000 kbps. Basándose en esta opción, FFmpeg calculará y corregirá el bitrate promedio producido [51].

`-map 0:a`: selecciona el flujo de audio, porque en nuestra implementación solo se requiere un mismo archivo de audio para todos los tamaños de vídeo.

`-use_template 1`: habilita el uso de SegmentTemplate en lugar de SegmentList. Así no se enumerará cada segmento en el manifiesto, sino que se usará una plantilla.

`-use_timeline 1`: habilita el uso de SegmentTimeline dentro de las SegmentTemplate. Permite especificar duraciones exactas de cada segmento en la plantilla.

`-seg_duration 4`: indica la duración en segundos de los segmentos, en este caso 4.

`-frag_duration 4`: indica la duración en segundos de los fragmentos dentro de los segmentos. En DASH es preferible alinear los fragmentos con los segmentos [34].

`-adaptation_sets "id=0, streams=v id=1,streams=a"`: asigna los flujos correspondientes a cada Adaptation Set.

`-f dash manifestffmpeg.mpd`: indica que se utilizará el formato dash. Se indica también el nombre del archivo MPD de salida.

Otras opciones útiles:

`-window_size 5`: indica el número de segmentos guardados en el manifiesto, es decir, en el MPD solamente se reflejan los últimos segmentos indicados, en este caso 5. El

reproductor podrá volver atrás, como máximo, el tiempo correspondiente al tamaño de ventana indicado. Es relevante para una transmisión en directo, donde quizá se pueda prescindir de los segmentos que los clientes ya habrán solicitado anteriormente. Hay que tener en cuenta que al finalizar la transmisión en directo, el MPD pasará a ser estático y se habrán perdido todos los segmentos anteriores [54].

`-extra_window_size 5`: indica el número de segmentos no guardados en el manifiesto pero almacenados en el disco duro antes de desaparecer [54]. En este proyecto se ha comprobado que se almacenan por defecto 5 segmentos.

`-rtbufsize 10M`: establece la máxima memoria en megabytes utilizada para almacenar frames a tiempo real. Ampliar este buffer puede evitar la pérdida de frames si llegan antes de finalizar el procesado de los anteriores, pero aumenta el retardo [57].

`-remove_at_exit 1`: exige la eliminación de todos los archivos al detener la emisión. En ese momento, desaparecen automáticamente los segmentos y el manifiesto [58].

`-tune zerolatency`: Exige una optimización para streaming con codificación rápida y baja latencia [59]. Durante la elaboración de este proyecto no se han apreciado cambios reseñables tras la utilización de esta opción, rondando un retardo de entre 10 y 15 segundos en ambos casos.

4.2.2.3. Conclusiones

FFmpeg es una herramienta muy completa, sencilla y rápida que consigue los mismos o mejores resultados que otros programas y con menos pasos.

Su documentación es notoriamente más extensa y añade muchas utilidades adicionales.

Además de contar con constantes actualizaciones, es fácil encontrar guías de uso actualizadas, pero también artículos independientes y comentarios en foros técnicos que demuestran que es una opción de uso amplio y extendido y una de las más buscadas.

Todas las herramientas han cumplido su cometido, pero sus múltiples funcionalidades y su facilidad de manejo hacen a FFmpeg una opción muy atractiva.

4.3. Alojamiento y transmisión

Para poder recibir contenido DASH que pueda reproducirse, hay que alojarlo en un servidor. Primero se mostrará cómo crear un servidor local con el que se pueda comprobar el funcionamiento de los archivos generados accediendo a ellos con mayor velocidad y sin necesidad de contratar un dominio, y posteriormente se procederá a crear un dominio web para alojar este contenido en la nube.

Adicionalmente, conoceremos dos herramientas ideadas para la limitación de la conexión, que ayudarán a simular un estado cambiante de la capacidad de la red.

4.3.1. Servidor local

Se va a establecer un servidor local mediante el uso de la aplicación JSON Server, opción elegida por su sencillez y la rapidez de su implementación.

JSON es un formato de intercambio de información entre sistemas informáticos mediante texto plano basada en el lenguaje Javascript. Es más moderno y simple que XML. Tienen una forma reconocible de declarar objetos, arrays y valores. Sus datos no tienen significado, no han de entenderlas otras aplicaciones, pero tiene librerías que permiten que sea utilizado por la mayoría de lenguajes de programación [60].

JSON permite crear una API con datos de prueba que imita a un servicio web [61].

Una API, una interfaz de programación de aplicaciones, es un conjunto de protocolos utilizados para desarrollar e integrar el software de las aplicaciones permitiendo su comunicación con otras mediante una serie de reglas. Son las normas con las que un módulo se comunicará con otro, aceptando o limitando sus permisos [62].

Utiliza un protocolo cliente/servidor sin estado en el que cada petición HTTP tiene toda la información que se necesite y los datos habitualmente se almacenan en la memoria RAM. Así, ni el cliente ni el servidor necesitan recordar estados previos. Recordar estados requiere memoria de almacenamiento. A veces se utiliza una memoria caché para que el cliente pueda ejecutar la misma respuesta para peticiones idénticas, pero no es necesariamente recomendable [63].

Es necesario instalar node.js. Se puede descargar el paquete aquí:

https://nodejs.org/en/

Con el siguiente comando, se instala el servidor JSON.

```
npm install –g json-server
```

En cualquier carpeta de nuestro PC se puede crear un archivo de prueba con extensión .json, aunque, si no se crea, se creará uno automáticamente. Este servirá de origen de datos para la API.

Tras crear la carpeta /public en la misma ruta, se ha de crear dentro de ella un archivo index.html donde se enlazarán de una forma u otra los distintos contenidos.

Ejecutamos el siguiente comando en el terminal desde la ubicación del archivo inicio.json.

```
json-server --watch inicio.json
```

Tras este paso, el servidor ya se encuentra en funcionamiento y es posible acceder a él desde la URL https://localhost:3000

Si la información presente en el archivo index.html no aparece en pantalla al acceder a la URL, puede que sea necesario crear este archivo mediante el siguiente comando [64] y volverlo a editar.

```
$ touch index.html
```

4.3.2. Servidor en la nube

A la hora de alojar nuestro contenido en la nube para poder disponer de él de forma remota, se ha optado trabajar con Firebase Hosting, que cuenta con una licencia gratuita y 10 GB de almacenamiento.

Firebase es un proveedor de hosting web estático y orientado a desarrolladores creado en 2012 y gestionado por Google desde 2014 [65]. Ha sido diseñado para aplicaciones front-end. Los archivos son alojados en SSD, discos de estado sólido con memoria no volátil, en servidores de borde de CDN de todo el mundo para estar lo más cerca posible del cliente y reducir la latencia con una conexión segura bajo el certificado SSL de seguridad [66].

Firebase tiene su propia consola de comandos. Se puede descargar el archivo aquí:

https://firebase.google.com/docs/hosting/

Es necesario iniciar un proyecto, elegir un directorio raíz y registrar la app [67]. El directorio raíz puede modificarse posteriormente editando el archivo firebase.json [68].

Todos los archivos que sea necesario alojar online han de colocarse en la carpeta /public. Este proceso de hosting se inicia ejecutando el siguiente comando desde la consola de Firebase, que se iniciará abriendo el archivo descargado.

```
firebase deploy
```

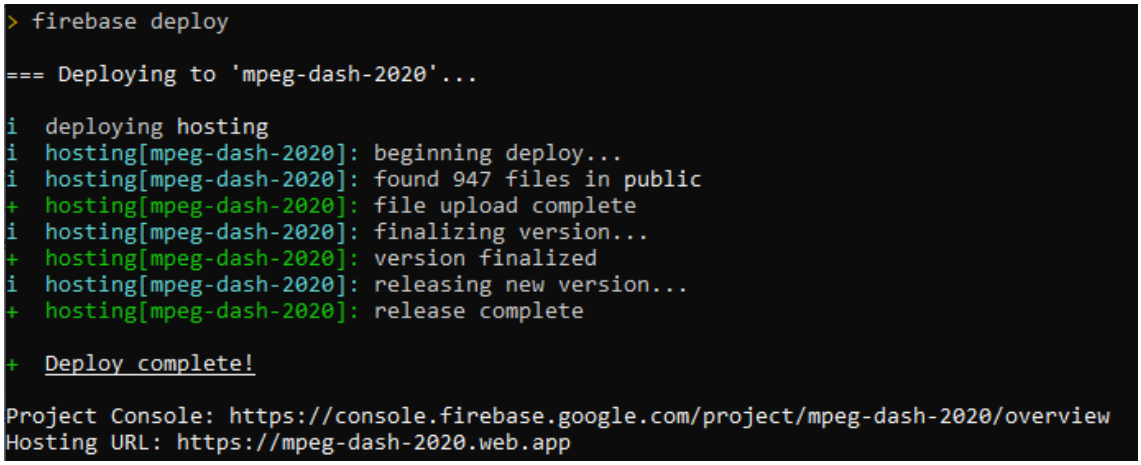

Figura 33: Proceso de deploy de Firebase

Tras este paso, el contenido ya se encuentra disponible en la URL https://mpeg-dash-2020.web.app, cuya designación se forma a partir del nombre elegido para el proyecto, y se puede acceder directamente a los archivos alojados allí añadiendo su nombre a la ruta. Por ejemplo, el MPD manifest.mpd se encontrará en https://mpeg-dash-2020.web.app/manifest.mpd.

4.3.3. Limitación de la velocidad de conexión

Existen herramientas y programas para limitar la velocidad de subida o bajada de datos y priorizar o garantizar ciertas conexiones. Esto puede servir para evitar que uno de nuestros dispositivos consuma gran parte del ancho de banda de nuestra conexión y limite el del resto. En este caso, ayudarán a fingir una conexión pobre a la que el cliente deberá adaptarse solicitando los archivos correspondientes.

4.3.2.1. Google Chrome

Chrome cuenta con una herramienta integrada capaz de limitar el ancho de banda que permite crear perfiles personalizados. Para utilizarla, hay que acceder al espacio de Herramientas para desarrolladores.

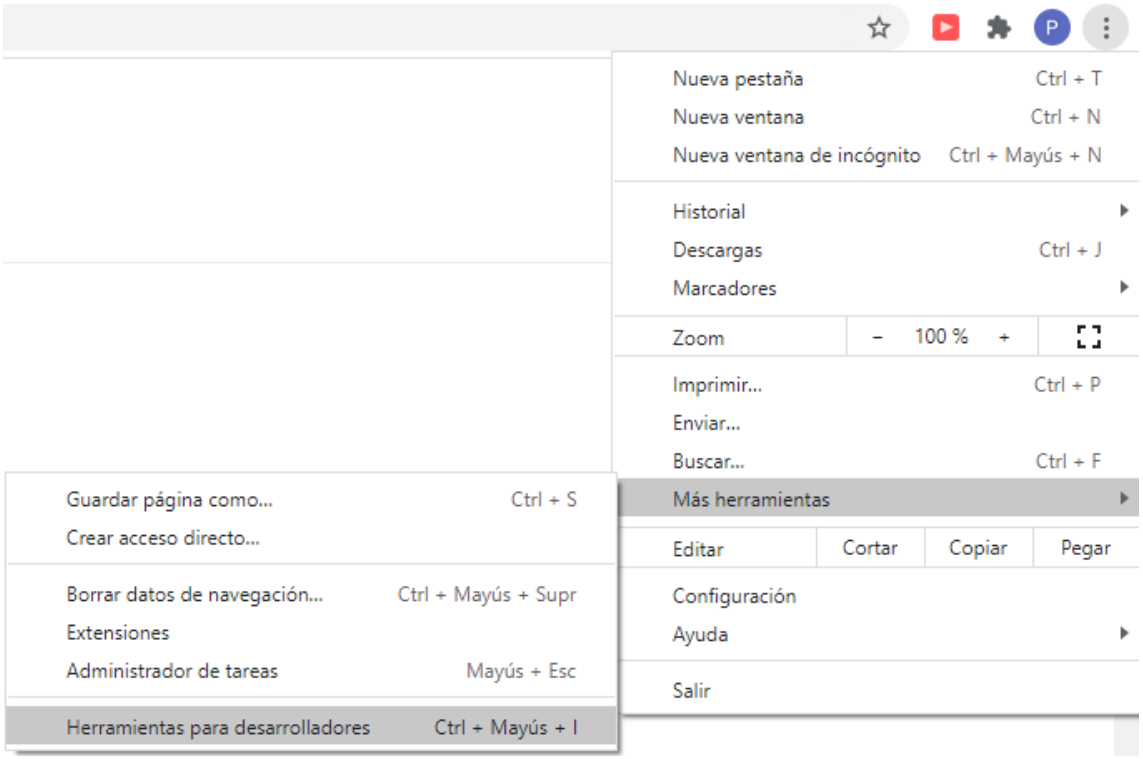

Figura 34: Acceso a Herramientas para desarrolladores de Google Chrome

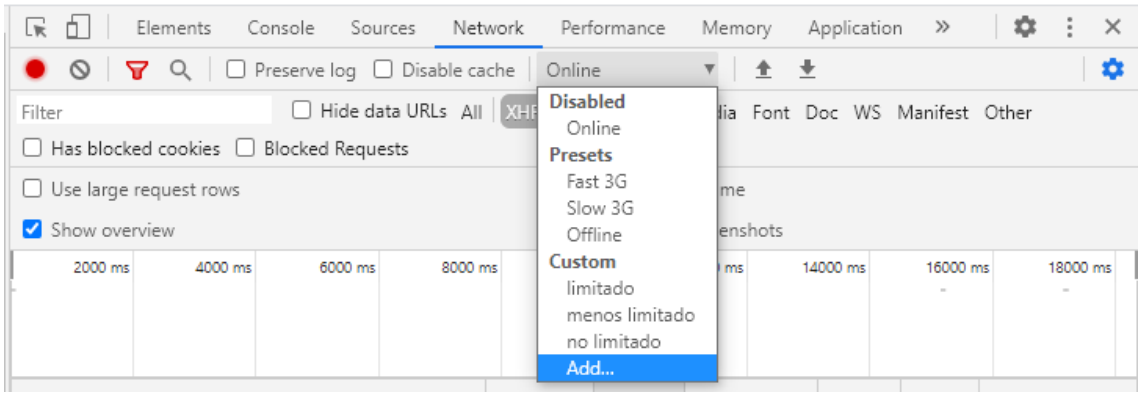

Figura 35: Acceso a la lista de perfiles personalizados de limitación de ancho de banda

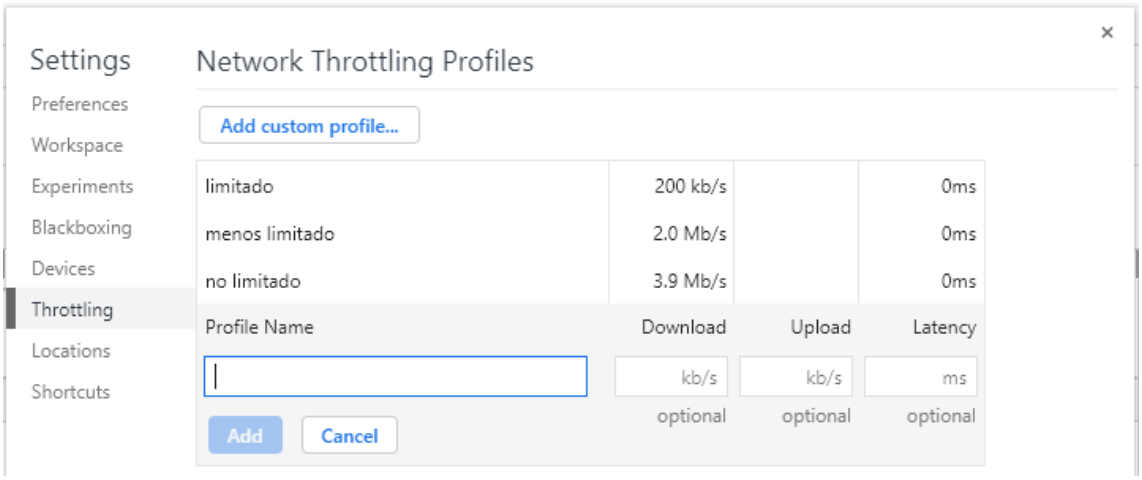

Figura 36: Añadir un perfil personalizado de limitación de ancho de banda

La conexión queda limitada automáticamente al seleccionar uno de los perfiles creados o ya existentes.

4.3.2.2. NetLimiter

NetLimiter es una herramienta desarrollada para Windows para proporcionar un control total de la red. Ha sido desarrollada por Locktime Software desde 2005 [69].

Es posible descargar el programa aquí:

https://www.netlimiter.com/

NetLimiter permite limitar el ancho de banda o la velocidad de descarga de toda la red o de un proceso concreto, como por ejemplo Google Chrome o VLC. Es muy visual y fácil de utilizar. Tiene 28 días de prueba.

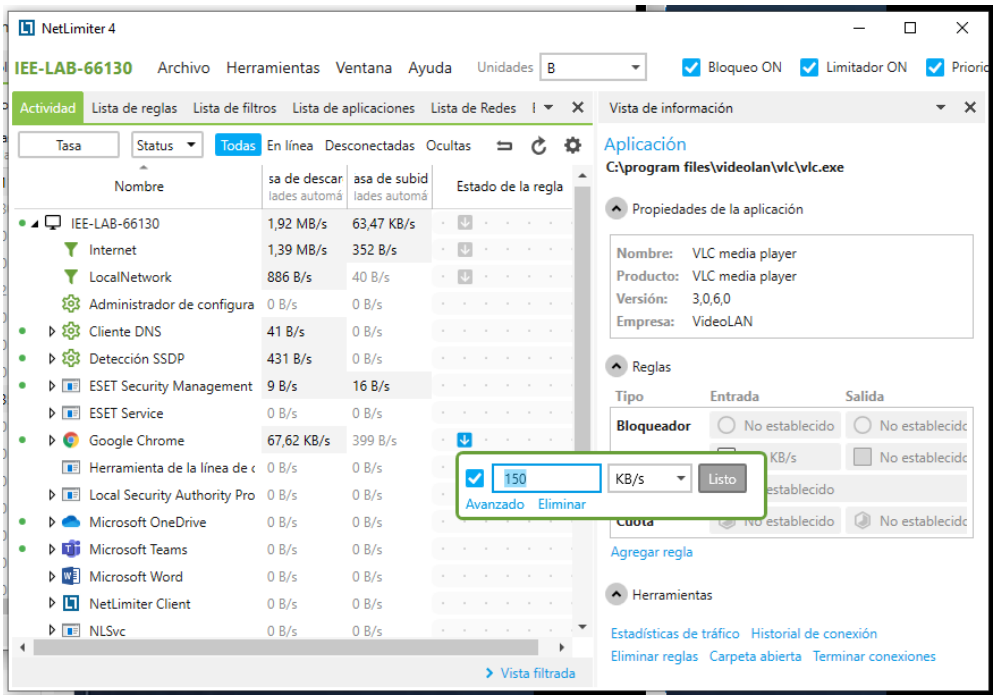

Figura 37: Inferfaz de NetLimiter

4.4. Reproducción

Después de crear contenido DASH y alojarlo en un servidor, será necesario comprobar si se reproduce correctamente. Es posible encontrar varios reproductores que cuentan con la implementación del estándar DASH y posibilitan la lectura de los archivos generados.

4.4.1. Dash.js

DASH Industry Forum es una compañía que busca la interoperabilidad de productos habilitados para DASH existentes en el mercado y, para ello, crea guías de interoperabilidad para el uso del estándar [70].

Su reproductor de referencia dash.js [71], de código abierto, fue el primero en utilizar su algoritmo de adaptación de bitrate BOLA, que minimiza el rebuffering (parones en mitad de la reproducción) y, a diferencia de las investigaciones anteriores, no depende de una predicción del ancho de banda disponible de la red, sino de la cantidad de datos almacenada en el buffer. Netflix ya utiliza este algoritmo. El tercer algoritmo disponible, llamado Dynamic, es un híbrido que al inicio emplea una simple estimación del ancho de banda y más adelante utiliza el BOLA [72].

El reproductor puede incrustarse fácilmente en una página web [73][74].

4.4.1.1. Uso

Cuenta con la ventaja de la selección de opciones en el momento, sin necesidad de modificar su código. Basta con introducir la URL del manifiesto y marcar las opciones deseadas antes de cargarlo.

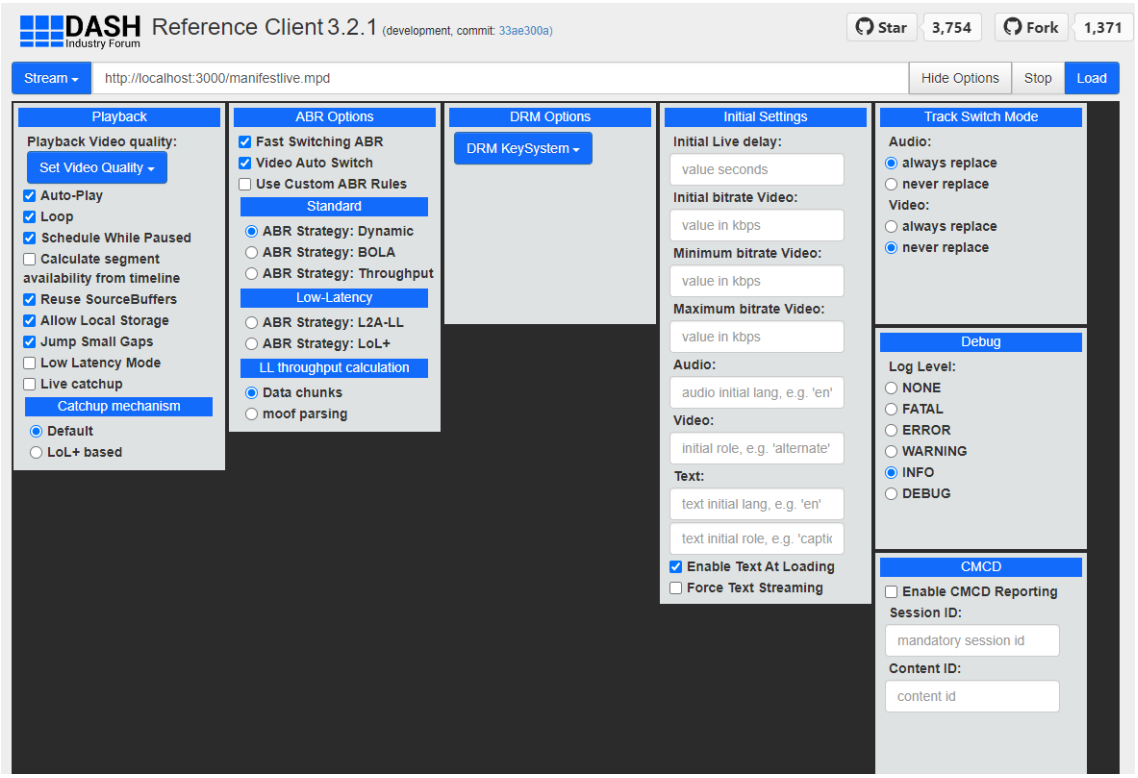

Figura 38: Opciones del reproductor de referencia de DASH-IF.

Al limitar el ancho de banda, el buffer comienza a vaciarse. Es entonces cuando el reproductor solicita segmentos de menor calidad y el buffer puede llenarse de nuevo.

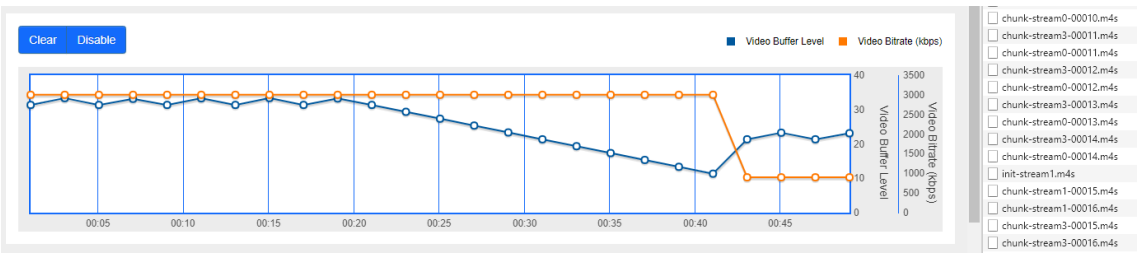

Figura 39: Cambio de Representation al vaciarse el buffer

Puede apreciarse en la parte derecha de la figura 39 que los segmentos solicitados al comienzo son los correspondientes a las Representations 0 (vídeo a 1920x1080 y 3000 kbps) y 3 (audio). A partir del vaciado del buffer, se solicita el segmento de inicialización de la Representation 1 (vídeo a 640x360 y 1000 kbps) y, a partir de ese momento, los segmentos correspondientes.

4.4.1.2. Conclusiones

Este reproductor es capaz de leer MPD estáticos y dinámicos alojados tanto online como en el servidor local y permite, entre otras opciones, indicar un bitrate de vídeo inicial. Es una buena opción para comprobar la validez de manifiestos de creación propia.

Durante la elaboración de este proyecto, las transiciones entre Representations han sido inmediatas en la transmisión de archivos bajo demanda, pero en transmisiones en directo, en las transiciones a Representations más bajas, se han dado parones de persistencia considerable que no se han solventado aumentando o disminuyendo la duración de los segmentos.

Cabe destacar que no reproduce el contenido alojado online con Firebase.

4.4.2. VLC

Es un software multiplataforma, libre y de código abierto que permite reproducir DVD, archivos comprimidos o vídeo procedente de una URL o una IP, es decir, permite ver vídeo en streaming. Consume pocos recursos en el sistema y soporta multitud de codecs. El proyecto comenzó a desarrollarse por VideoLAN en 1996 y fue lanzado en 2001 [75].

Permite la reproducción de contenido DASH al menos desde su versión 3.0.0 [76].

4.4.2.1. Uso

No es posible abrir los MPD como si fuesen archivos de vídeo, pero sí se pueden reproducir desde una URL.

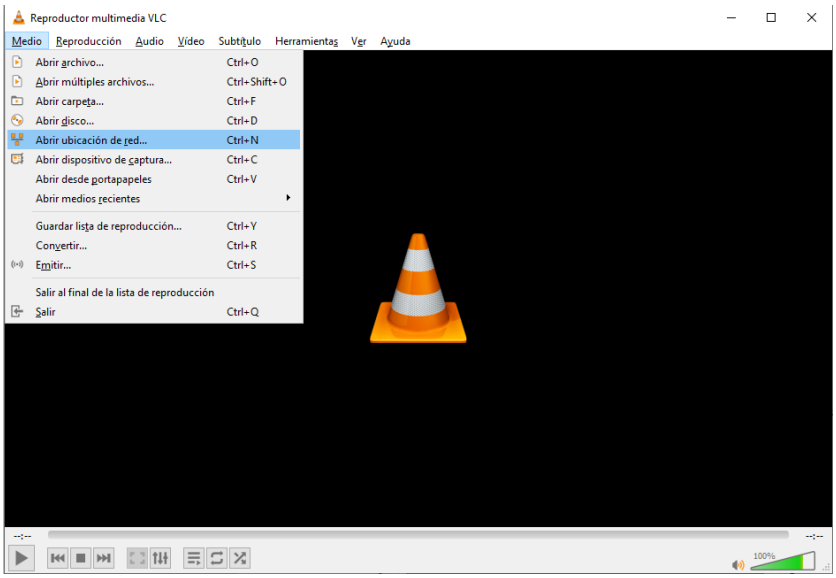

Figura 40: Abrir ubicación de red en VLC

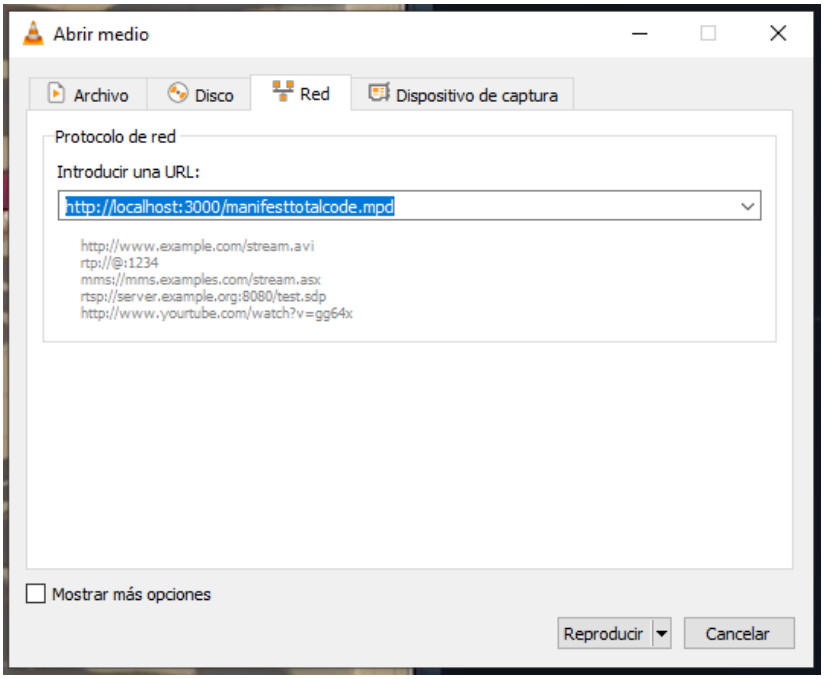

Figura 41: Introducir la URL del MPD

4.4.2.2. Conclusiones

VLC siempre ha sido siempre una herramienta útil y accesible. Su implementación de DASH es básica y, aunque quizá es demasiado limitada para desarrolladores, la reproducción de contenido es más que suficiente.

Además, reproduce correctamente el contenido alojado online con Firebase tanto en el PC como en su app móvil. En cambio, no soporta la señalización con rangos de bytes en transmisiones bajo demanda.

4.4.3. Extensiones de Chrome y Firefox

Las extensiones de MPEG-DASH permiten al navegador reproducir contenido DASH nativamente. En constante actualización, sus primeras versiones estuvieron disponibles en 2016 [77][78].

4.4.1.1. Uso

Basta con clicar en un enlace a un manifiesto DASH o introducirlo en la barra de búsqueda para reproducirlo directamente. Por defecto, el navegador descarga los archivos solicitados, el plugin comprueba si coinciden y abre una nueva pestaña con un reproductor de vídeo que utiliza la librería dashjs.

Figura 42: Primera Representation de vídeo bajo demanda en Chrome

Figura 43: Tercera Representation de vídeo bajo demanda en Chrome

Figura 44: Transmisión de un archivo de vídeo en Chrome con conversión en vivo

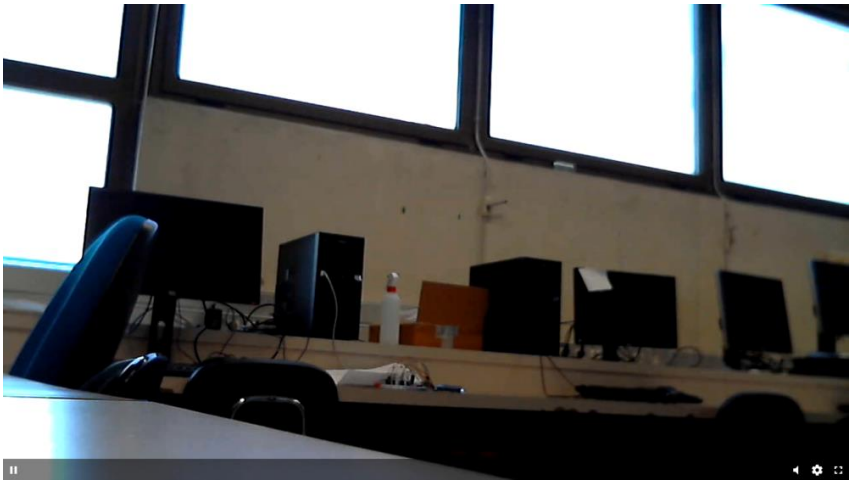

Figura 45: Transmisión en directo en Chrome

4.4.1.2. Conclusiones

En una transmisión en directo o en una conversión en vivo no se permite el time-shifting (retroceso en la línea de tiempo o pausa). Como se puede apreciar en las figuras 44 y 45, la barra de tiempo desaparece.

Durante la elaboración de este proyecto, las transiciones entre Representations han sido inmediatas en la transmisión de archivos bajo demanda, pero en transmisiones en directo, en las transiciones a Representations más bajas, se han dado parones de persistencia considerable (alrededor de 10 segundos) que no se han solventado aumentando o disminuyendo la duración de los segmentos.

No se han apreciado grandes diferencias entre el rendimiento del reproductor de referencia de DASH-IF y las extensiones de reproducción nativa de Chrome y Firefox en cuanto a la latencia de una transmisión en directo, de entre 10 y 15 segundos. Como es lógico, dados los parones antes mencionados, la reproducción se aleja cada vez más del directo en cada transición entre Representations en ambas plataformas.

Las extensiones también reproducen el contenido alojado online con Firebase.

4.4.4. Bitmovin

Bitmovin es una empresa tecnológica austriaca que provee servicios de codificación de vídeo y audio a formatos de streaming y contribuye con MPEG-DASH como autor de librerías y desarrollador de codificadores, encriptadores, reproductores y otras herramientas para desarrolladores. Fue fundada en 2013 [79].

4.4.4.1. Uso en PC

El reproductor de Bitmovin se incrusta fácilmente en una web añadiendo el código HTML y Javascript correspondiente a su SDK [80]. Actualmente tiene un periodo de prueba de 30 días.

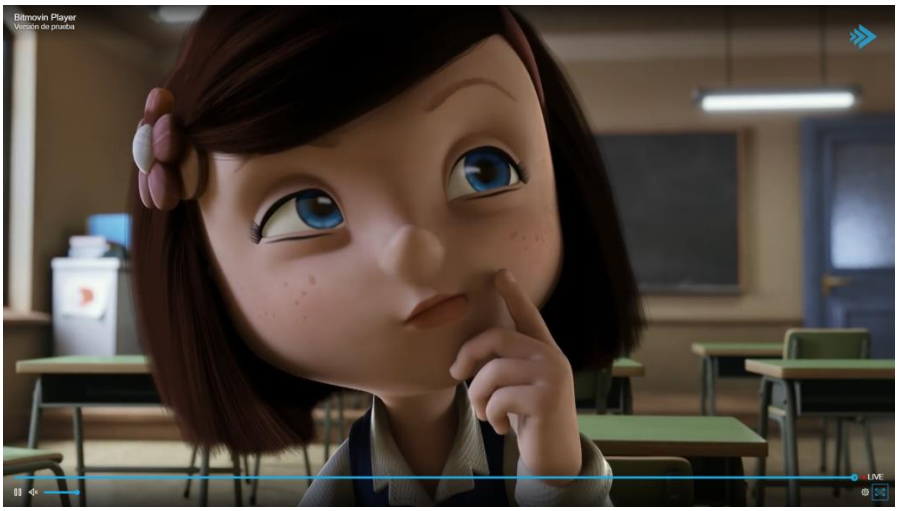

Figura 46: Transmisión de un archivo de vídeo en Bitmovin con conversión en vivo

Figura 47: Time-shifting en transmisión de vídeo en Bitmovin con conversión en vivo

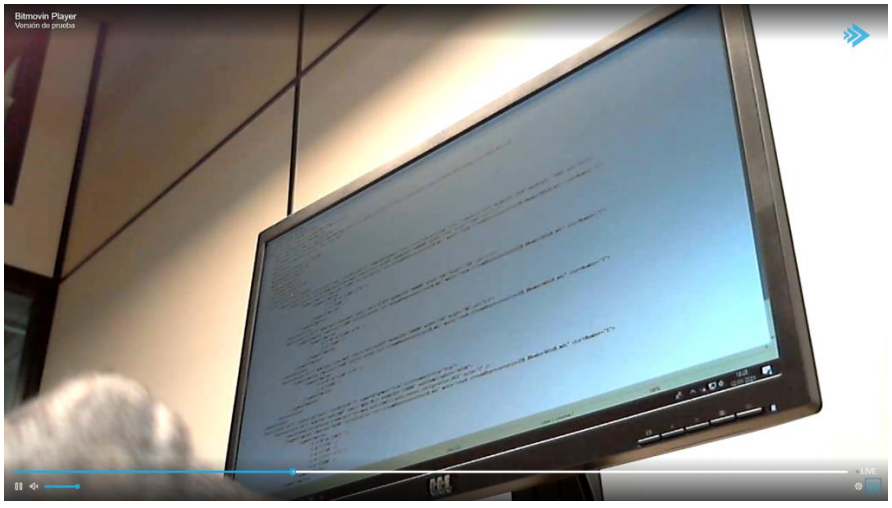

Figura 48: Time-shifting en una transmisión en directo en Bitmovin

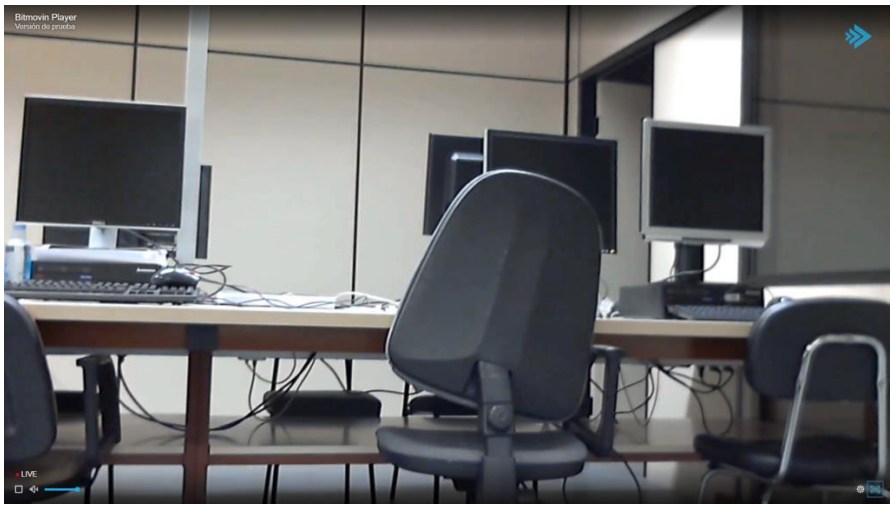

Figura 49. Time-shifting con tamaño limitado de ventana en directo en Bitmovin

4.4.4.2. Uso en Android

Es necesario instalar la aplicacion Bitmovin Player que se encuentra en la Play Store.

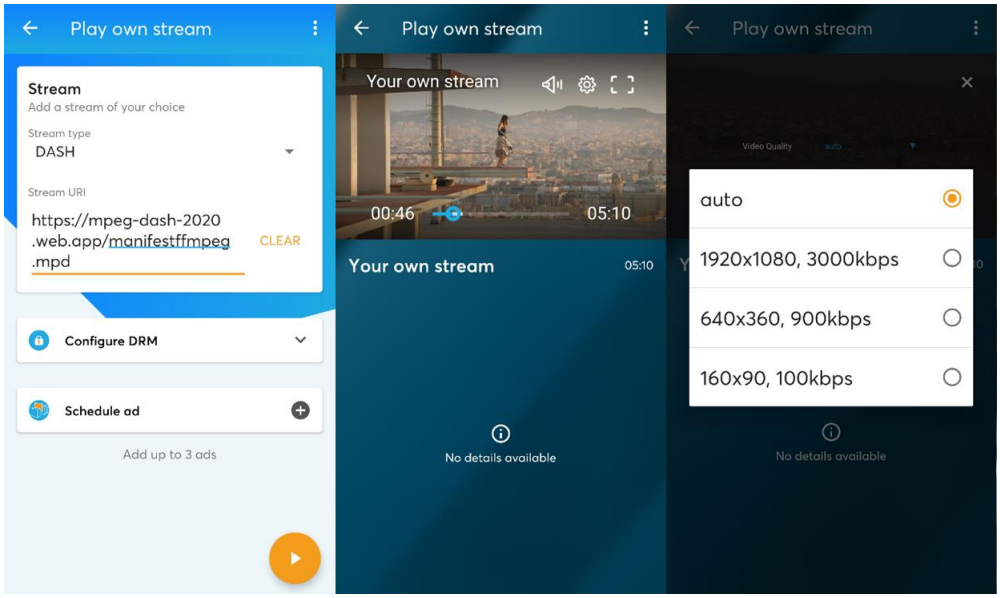

Figura 50: Transmisión bajo demanda en la app móvil de Bitmovin

Tras abrir la app, basta con seleccionar la opción *Own stream*, elegir DASH en el apartado *Stream type* e indicar el URI donde se aloja nuestro MPD.

Como puede verse en la figura 50, permite la configuración de DRM y la programación de anuncios.

4.4.4.3. Conclusiones

El reproductor de Bitmovin sí permite el time-shifting (retroceso en la línea de tiempo o pausa) en transmisiones en directo y en conversiones en vivo, pero no parece compatible con la opción de tamaño de ventana limitado (window_size). Cuando se incluye esta opción en FFmpeg y, por consiguiente, solo se almacenan en el manifiesto unos cuantos segmentos, el reproductor refleja que es una transmisión en directo pero, como se puede apreciar en la figura 49, la barra de tiempo desaparece.

Durante la elaboración de este proyecto, las transiciones entre Representations han sido inmediatas en la transmisión de archivos bajo demanda. En cuanto a las transmisiones en directo, en las transiciones a Representations más bajas, se han dado parones de alrededor de 2 segundos, una duración notablemente menor que las de los parones sufridos en el reproductor dash.js y las extensiones de Chrome y Firefox.

En comparación con estos reproductores mencionados, también la latencia en transmisiones en directo es considerablemente menor, concretamente de unos 6 segundos en el mejor de los casos. Los parones durante la transición entre Representations aumentan la latencia, pero también en menor medida.

Es bastante completo y fácil de implementar y utilizar y sus ventajas son palpables. Su aplicación móvil es un punto a favor a la hora de elegirlo. Que su uso sea de pago no es de extrañar. Su app, aunque de interfaz simple y utilidades básicas, es gratuita.

Cabe destacar que, a diferencia de la aplicación móvil, el reproductor web no lee el contenido alojado online con Firebase. Por otro lado, no soporta señalización con rangos de bytes en transmisiones bajo demanda en ninguna de las dos opciones.

4.4.5. NexPlayer HTML5

NexPlayer es un reproductor lanzado en 2017 [81] por la empresa NexStreaming, fundada en 2002 y con sede en Corea del Sur [82]. Soporta MPEG-DASH desde su creación.

Ha sido verificado por Unity para garantizar la compatibilidad con su editor y, aunque no posee su propia app, tiene SDK compatible con iOS y Android [83].

Actualmente tiene un periodo de prueba de 15 días.

4.4.5.1. Uso

El reproductor NexPlayer se incrusta fácilmente en una web añadiendo el código HTML y Javascript correspondiente a su interfaz [84].

Figura 51: Transmisión de un archivo de vídeo bajo demanda en NexPlayer

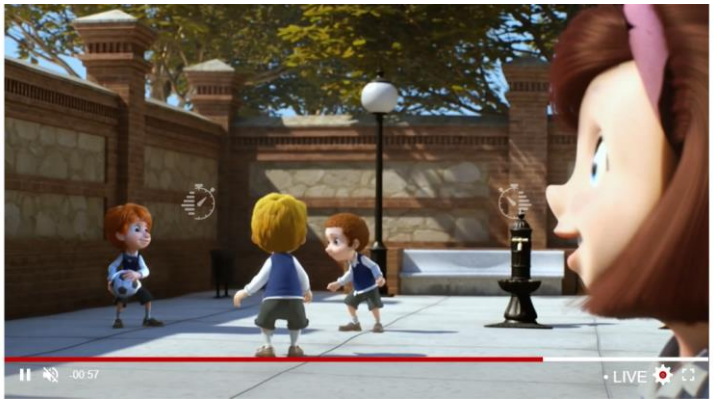

Figura 52: Time-shifting en transmisión de vídeo en NexPlayer con conversión en vivo

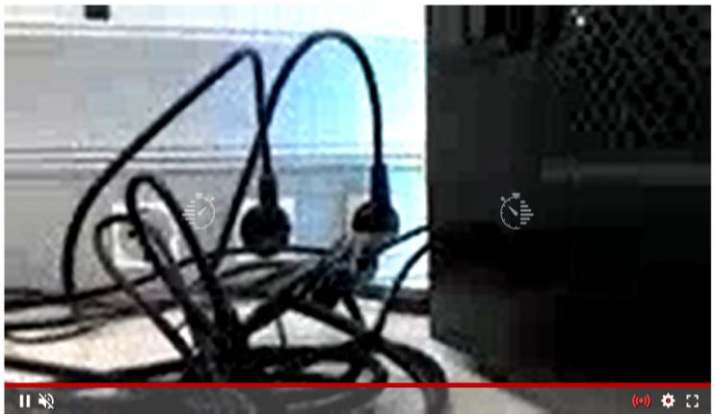

Figura 53: Transmisión en directo en NexPlayer

Figura 54: Time-shifting en una transmisión en directo en NexPlayer:

Figura 55: Time-shifting limitado a 16 segundos en directo en NexPlayer

4.4.5.2. Conclusiones

El reproductor NexPlayer también permite el time-shifting y, además, es compatible con la opción de tamaño de ventana limitado (window_size). Como se puede apreciar en la figura 55, el reproductor es capaz de retroceder en la reproducción tanto tiempo como se haya establecido.

Durante la elaboración de este proyecto, las transiciones entre Representations han sido inmediatas en la transmisión de archivos bajo demanda y han sufrido parones de alrededor de 2 segundos en las transiciones entre Representations, cifras comparables a las del reproductor de Bitmovin.

La latencia en las transmisiones en directo es también relativamente pequeña (8 segundos en el mejor de los casos) y aumenta igualmente en menor medida que en otros reproductores entre las transiciones.

Es de implementación y uso fácil y sus ventajas son comparables a la del reproductor Bitmovin. Sin duda, son dos buenas opciones y es comprensible que no sean gratuitas.

Cabe destacar que no reproduce el contenido alojado online con Firebase.

4.5. Otras herramientas utilizadas durante el proyecto

4.5.1. Microsoft Teams

Teams es un área de trabajo en equipo de la línea de servicios Microsoft 365 de Microsoft que integra usuarios, contenido y herramientas. Incluye conversaciones de chat, reuniones de vídeo, almacenamiento de archivos colaborativos y aplicaciones integradas. Fue lanzado en 2017 y es gratuito desde 2018 pero limitando el número de usuarios y la capacidad de almacenamiento [85].

La gestión del proyecto se ha organizado, programado y monitorizado con las herramientas propias de la aplicación. Las reuniones semanales se planificaron y marcaron en Teams, las tareas sugeridas por el director del TFG o autoimpuestas que facilitaban el avance del proyecto se publicaron gracias a la herramienta Planner y la comunicación entre las partes se realizó mediante la sección de publicaciones. Por último, todos los archivos creados y utilizados durante la elaboración del proyecto se alojaron en una carpeta de SharePoint Server mediante una suscripción educativa a Microsoft 365 de la Universidad Pública de Navarra [86].

Teams puede descargarse aquí:

https://www.microsoft.com/es-es/microsoft-teams/download-app

4.5.2. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsot y lanzado en 2015. Es gratuito y de código abierto. Incluye soporte para depuración, control integrado de Git, resaltado de sintaxis y finalización inteligente de código, y es compatible con multitud de lenguajes de programación [87].

Todas estas funcionalidades, previa edición de directivas de ejecución [88], permitieron facilitar la edición, la corrección y la ejecución de códigos en el proyecto.

Visual Studio Code puede descargarse aquí:

https://code.visualstudio.com/

4.5.3. HTTrack

HTTrack es una aplicación libre y de código abierto que permite descargar todo o parte del contenido de un sitio web [89].

Este programa facilitó la descarga de manifiestos y segmentos ajenos desde webs de referencia para su estudio posterior. Es muy sencillo pero efectivo.

HTTrack puede descargarse aquí:

http://www.httrack.com/page/2/

5. Conclusiones y líneas abiertas

5.1. Conclusiones

Las plataformas de vídeo se adaptan a la calidad de la conexión pero no la controlan, ya que no gestionan las redes sobre las que se apoyan. Por lo tanto, no pueden asegurar una calidad de servicio aceptable. Sin embargo, el uso de redes ya existentes permite la llegada de sus servicios a un mayor número de gente.

La utilización de MPEG-DASH, como estándar universal que reúne las mejores características de otros protocolos propietarios, facilita la implementación y la integración en distintos sistemas audiovisuales y la compatibilidad con distintos dispositivos.

Se dispone ya de multitud de herramientas para crear y reproducir contenido MPEG-DASH, convirtiéndolo en una técnica atractiva y accesible.

Con la elección de las opciones adecuadas, es perfectamente posible generar, desde cero, contenido compatible con el estándar, alojarlo en la nube y reproducirlo con éxito. En este proyecto se ha conseguido crear, transmitir y reproducir contenido MPEG-DASH, pudiendo, además, comprobar las ventajas del estándar que ya utilizan las principales plataformas.

Las instrucciones indicadas aquí sientan unas bases que pueden suponer un punto de inicio en la especialización de esta técnica. También se han aportado diversas funcionalidades adicionales extraídas de los extensos manuales de usuario que aporta cada programa y atractivas para ahondar en la complejidad de la programación propia de cada elemento. Sin embargo, el estándar se extiende mucho más de lo aportado aquí.

Será tarea del desarrollador elegir, de entre todos los marcados, un camino de ejecución para lograr sus objetivos.

5.1.1. Tamaño de los segmentos

Existen multitud de estudios sobre la importancia del tamaño de los segmentos en una transmisión de contenido DASH. En ellos se concluye que los segmentos de una duración menor a 2 segundos deberían descartarse. Segmentos demasiado pequeños pueden ocasionar un rendimiento bajo de la transmisión debido a la sobrecarga producida por las peticiones y a los retardos de la red. Con segmentos grandes, pueden darse parones si se utilizan conexiones inalámbricas con muchos cambios de ancho de banda. Además, las transiciones entre Representations no se darían lo antes posible. El tamaño óptimo sería de de entre 5 y 8 segundos con conexiones no persistentes y de entre 2 y 3 segundos con conexiones persistentes.

Bitmovin recomienda segmentos de entre 2 y 4 segundos, que suponen un buen equilibrio entre la eficiencia de codificación y la flexibilidad de adaptación a los cambios de ancho de banda. Aumenta la importancia de la latencia en emisiones en directo [20].

En este proyecto, tras realizar pruebas de comprobación de resultados, se decidió establecer un tamaño de segmento de 4 segundos.

5.1.2. La importancia del tamaño del GOP

Dado que los reproductores pueden comenzar la reproducción solamente desde frames I, si el GOP es demasiado grande, estos frames estarán demasiado distanciados y el usuario no podrá acceder a un punto del contenido situado entre ellos. En su lugar, el reproductor saltará hasta el frame I más cercano. Esto puede ser indeseable si es importante la precisión de la búsqueda [90].

Precisamente esto fue determinante para decidir establecer un tamaño de GOP menor que el del segmento. En una gran cantidad de artículos consultados inicialmente para la elaboración de este proyecto se utilizan segmentos del mismo tamaño que el GOP [91] y, aunque otros desaconsejan utilizar tamaños de GOP pequeños por el incremento de información utilizada [92], en este proyecto se han detectado grandes mejoras en la reproducción y en la transición entre Representations estableciendo GOP de 25 frames con segmentos de una duración de 4 segundos (100 frames), es decir, los segmentos son del tamaño recomendado pero se establece el uso de más de un frame I dentro de ellos. Colocando solamente un frame I en cada segmento (duración del segmento equivalente al tamaño del GOP), la transición entre Representations no se realiza correctamente en la mayoría de los casos.

5.2. Líneas abiertas

5.2.1. HLS y baja latencia

HLS, abreviatura HTTP Live Streaming, es un protocolo de streaming adaptativo desarrollado originalmente por Apple para permitir al iPhone acceder a transmisiones en directo. Es poco probable que los firewalls bloqueen su contenido. Al principio, HLS era exclusivo de los iPhone, pero hoy casi todos los dispositivos lo soportan. Por ejemplo, HBO España; HBO Max, el servicio de streaming de la cadena HBO en Estados Unidos [93]; y Disney+ [94] utilizan HLS.

Puede reproducir vídeo codificado con H.264 o H.265. Recomienda segmentos de 10 segundos con un IDR cada 2 segundos, es decir, GOP cerrados.

Su latencia de entrega tiende a rondar los 45 segundos. Para evitar esto, pueden crearse segmentos más pequeños, pero suelen provocar aumentos en los requerimientos de caché del servidor y causar errores de reproducción o empeorar su calidad. Tal vez el reproductor no consiga entregar la imagen en 4K o se aprecien glitches debido a la pérdida de paquetes. Necesita mayor capacidad de CPU.

Se ha conseguido reducir la latencia a 15 segundos o menos con su nuevo método de baja latencia, que corta los segmentos en trozos más pequeños que pueden enviarse antes de que el segmento esté disponible completamente [95].

No existe gran diferencia entre la calidad de imagen que entregan MPEG-DASH o HLS. MPEG-DASH podía entregar mejor calidad con bitrate bajo y mejor resolución, pero esto ya no es así desde que HLS soporta H.265 y 4K. Ambos soportan HDR para mayores gamas cromáticas y mejor interpretación tonal. Ambos estándares son estables, fáciles de implementar y potentes.

Safari, el navegador por defecto de iPhone, iPad y Apple TV, no soporta MPEG-DASH, pero cualquier dispositivo de Android, iOS, Windows, Mac, Linux o Chrome OS soporta HLS. Existen más de mil millones de usuarios de iOS en todo el mundo que no pueden reproducir MPEG-DASH en sus navegadores habituales [96], aunque el reproductor NexPlayer ya reproduce DASH en iOS [97].

MPEG-DASH también tiene un método de baja latencia, conocido como LL-DASH [98]. Sus primeras demostraciones se expusieron en 2018 y sus guías de uso se han presentado en 2020 [99], pero el reproductor NexPlayer, por ejemplo, ya soporta esta técnica [97].

Un estudio de comparación entre HLS y MPEG-DASH podría servir como vía de ampliación de este proyecto.

5.2.2. H.265 y VP9

H.265, el sucesor de H.264, proporciona mejor calidad de vídeo con la misma tasa de datos, o entre un 25 y un 50 % de compresión adicional para la misma calidad de imagen. Es compatible con servicios de televisión en ultraalta definición y resoluciones hasta 8192x4320 [100].

VP9 es un formato de codificación abierto desarrollado por Google desde 2011. También dobla el factor de compresión de H.264 y soporta 8K. Es más rápido y eficiente en la codificación que H.265 [101].

Sería interesante investigar el rendimiento de VP9 y H.265 en el uso de MPEG-DASH e incluso una comparación entre ellos.

5.2.3. Subtítulos, miniaturas y encriptación

MPEG-DASH permite, entre otras muchas funcionalidades, la incrustación de subtítulos sobre el vídeo y la utilización de miniaturas para apoyar visualmente el avance rápido durante el contenido. Podría ser de interés hacer uso de estas técnicas para completar la experiencia.

Además, es posible encriptar el contenido. Sería necesario integrar un sistema DRM de código abierto que permitiese cifrar la información y gestionar claves para las licencias.

5.2.4. MP4Box sin x264

Como se ha comentado anteriormente, MP4Box ofrece la posibilidad de recodificar el contenido sin la ayuda de otros programas desde finales del año 2020. Sería interesante comprobar su eficacia y comparar los resultados con los ofrecidos por FFmpeg.

5.2.5. Hosting de una transmisión en directo

En este proyecto, las pruebas con emisiones en directo por webcam se han realizado siempre desde el servidor local.

Firebase precisa del proceso de deploy para que su contenido se encuentre accesible, lo que dificulta la emisión en directo.

Otra línea abierta consistiría en la búsqueda de plataformas de hosting automático del contenido generado en vivo para poder disponer de él también de forma remota.

6. Bibliografía

[1] Sandvine publica el 2019 Global Internet Phenomena Report, 10 de septiembre de 2019. [En línea]. Disponible en: https://www.sandvine.com/press-releases/sandvine-releases-2019-global-internet-phenomena-report

[2] Netflix supera los 200 millones de usuarios. [En línea]. Disponible en: https://www.hollywoodreporter.com/news/netflix-tops-200-million-subscribers-amid-pandemic

[3] Cuántos suscriptores tiene Netflix, HBO, Disney+, Apple TV+ y Amazon Prime Video, 11 de septiembre de 2020. [En línea]. Disponible en: https://www.revistagq.com/noticias/articulo/cuantos-suscriptores-tiene-netflix-hbo-disney-apple-amazon-prime-video

[4] Estrategias de marketing de vídeo, 16 de diciembre de 2019. [En línea]. Disponible en: https://www.vidyard.com/blog/video-marketing-strategy/#measure-your-performance

[5] Netflix ve ahorro de costes en la adopción de MPEG DASH, 15 de diciembre de 2011. [En línea]. Disponible en: https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=79409

[6] Entregar contenido en vivo en YouTube vía DASH. [En línea]. Disponible en: https://developers.google.com/youtube/v3/live/guides/encoding-with-dash

[7] Twitch vs. Amazon Prime para el fútbol de los jueves por la noche, 15 de noviembre de 2018. [En línea]. Disponible en: https://mux.com/blog/thursday-night-football-streaming-technology-showdown-amazon-prime-vs-twitch/

[8] Moving Picture Experts Group, Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Moving_Picture_Experts_Group

[9] MPEG-DASH, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP

[10] Guía de implementación de DASH-IF: Puntos de interoperabilidad, 7 de agosto de 2019. [En línea]. Disponible en: https://dashif-documents.azurewebsites.net/DASH-IF-IOP/pull/341/DASH-IF-IOP.pdf

[11] El estándar MPEG-DASH para streaming multimedia sobre la Internet, octubre-diciembre de 2011. [En línea]. Disponible en: https://www.bogotobogo.com/VideoStreaming/images/mpeg_dash/T_MM1_TheMPEGDASHStandard.pdf

[12] Guía de implementación de DASH-IF: Puntos de interoperabilidad, 9 de abril de 2018. [En línea]. Disponible en: https://dashif.org/docs/DASH-IF-IOP-v4.2-clean.pdf

[13] La estructura de un MPD en MPEG-DASH, 20 de marzo de 2015. [En línea]. Disponible en: https://www.brendanlong.com/the-structure-of-an-mpeg-dash-mpd.html

[14] XML Namespaces. [En línea]. Disponible en: https://www.w3schools.com/xml/xml_namespaces.asp

[15] DASH-IF: Perfiles. [En línea]. Disponible en: https://dashif.org/identifiers/profiles/

[16] Empaquetado de MPEG-DASH. Perfiles. [En línea]. Disponible en: https://docs.unified-streaming.com/documentation/package/mpeg-dash.html

[17] Guía de implementación de DASH-IF: Modelo de timing restringido, 12 de junio de 2020. [En línea]. Disponible en: https://dashif-documents.azurewebsites.net/Guidelines-TimingModel/master/Guidelines-TimingModel.pdf

[18] Guía de implementación de DASH IF: Puntos de interoperabilidad, 25 de septiembre de 2019. [En línea]. Disponible en: https://dashif-documents.azurewebsites.net/DASH-IF-IOP/master/DASH-IF-IOP.html

[19] MPEG-DASH, 21 de abril de 2015. [En línea]. Disponible en: https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/

[20] Tamaño de los segmentos para un streaming óptimo, 9 de abril de 2015. [En línea]. Disponible en: https://bitmovin.com/mpeg-dash-hls-segment-length/

[21] GOP abiertos y cerrados, 23 de diciembre de 2019. [En línea]. Disponible en: https://streaminglearningcenter.com/articles/open-and-closed-gops-all-you-need-to-know.html

[22] Forzar la base de tiempo del vídeo para tener al menos una precisión de 0,1 ms, 7 de febrero de 2013. [En línea]. Disponible en: https://lists.ffmpeg.org/pipermail/ffmpeg-devel/2013-February/138640.html

[23] Configurar el multiplexador para no recalcular la base de tiempo en base a un valor absurdo de escala de tiempo, 6 de enero de 2018. [En línea]. Disponible en: https://lists.ffmpeg.org/pipermail/libav-user/2018-January/010843.html

[24] ¿Qué es timescale, timebase o timestamp en FFmpeg?, 10 de abril de 2017. [En línea]. Disponible en: https://stackoverflow.com/questions/43333542/what-is-video-timescale-timebase-or-timestamp-in-ffmpeg

[25] Atributo Duration en Segment Template. [En línea]. Disponible en: https://docs.aws.amazon.com/mediapackage/latest/ug/segtemp-format-duration.html

[26] DASH-IF: Metadatos de la fuente de audio. [En línea]. Disponible en: https://dashif.org/identifiers/audio_source_metadata/

[27] Entender y configurar un MPD dinámico. [En línea]. Disponible en: https://docs.unified-streaming.com/documentation/live/configure-dynamic-mpd.html

[28] H.264, Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/H.264/MPEG-4_AVC

[29] MainConcept, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/MainConcept

[30] El proyecto GPAC, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/GPAC_Project_on_Advanced_Content

[31] x264, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/X264

[32] Generación de contenido MPEG-DASH con MP4Box y x264, 3 de noviembre de 2014. [En línea]. Disponible en: https://bitmovin.com/mp4box-dash-content-generation-x264/

[33] Flujos multi-bitrate bajo demanda con MP4Box. [En línea]. Disponible en: https://www.radiantmediaplayer.com/guides/working-with-mp4box.html

[34] MPEG-DASH: Fragmentación y segmentación, 3 de diciembre de 2018. [En línea]. Disponible en: https://stackoverflow.com/questions/49516434/mpeg-dash-fragmentation-and-segmentation

[35] Comandos MP4Box. [En línea]. Disponible en: https://www.mankier.com/1/mp4box

[36] Usar MP4Box para cambiar la resolución, 16 de marzo de 2019. [En línea]. Disponible en: https://forum.videohelp.com/threads/392564-Using-MyMP4Box-to-change-resolution

[37] MP4Box: Empaquetado de contenido, 8 de diciembre de 2020. [En línea]. Disponible en: https://github.com/gpac/gpac/wiki/MP4Box

[38] Crear diferentes cualidades para mi manifiesto con solo MP4Box, 29 de septiembre de 2020. [En línea]. Disponible en: https://stackoverflow.com/questions/64057660/create-different-qaulities-for-my-manifest-with-only-mp4box

[39] Usando la nueva multicodificación de MP4Box para crear un MPD no multiplexado, 3 de octubre de 2020. [En línea]. Disponible en: https://github.com/gpac/gpac/issues/1603

[40] Bucle en dash en vivo con un archivo que crece, 1 de junio de 2017. [En línea]. Disponible en: https://github.com/gpac/gpac/issues/835

[41] Soporte DASH en GPAC. Perfiles, 21 de agosto de 2014. [En línea]. Disponible en: http://www.gpac-licensing.com/2014/08/21/ibc-2014-dash-avc264-support-in-gpac/

[42] FFmpeg, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/FFmpeg

[43] Sobrescribir ficheros en FFmpeg. [En línea]. Disponible en: https://www.xspdf.com/resolution/51576278.html

[44] Documentación FFmpeg. [En línea]. Disponible en: https://ffmpeg.org/ffmpeg.html

[45] MPEG-DASH solo con FFmpeg, 22 de abril de 2018. [En línea]. Disponible en: https://blog.infireal.com/2018/04/mpeg-dash-with-only-ffmpeg/

[46] Trabajar con FFmpeg. [En línea]. Disponible en: https://www.radiantmediaplayer.com/guides/working-with-ffmpeg.html#ffmpeg-h264

[47] Guía de opciones de x264 en FFmpeg. [En línea]. Disponible en: https://sites.google.com/site/linuxencoding/x264-ffmpeg-mapping

[48] Solución error librería libvo_aacenc, 9 de mayo de 2018. [En línea]. Disponible en: https://www.enmimaquinafunciona.com/pregunta/152736/desconocido-encoder-libvo_aacenc-error

[49] YUV, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/YUV

[50] Error ocurrido en –pix_fmt en FFmpeg convirtiendo de .mov a archivo mp4, 12 de septiembre de 2014. [En línea]. Disponible en: https://stackoverflow.com/questions/24799368/error-occured-on-pix-fmt-in-ffmpeg-while-converting-from-mov-to-mp4-file

[51] Limitar el bitrate de salida con FFmpeg, 2018. [En línea]. Disponible en: https://trac.ffmpeg.org/wiki/Limiting%20the%20output%20bitrate

[52] FFmpeg - Detener proceso si el archivo de salida ya existe, 2 de junio de 2015. [En línea]. Disponible en: https://superuser.com/questions/922866/ffmpeg-skip-process-if-output-already-exists

[53] Tee muxer sobreescribe los archivos de salida silenciosamente, enero de 2020. [En línea]. Disponible en: https://trac.ffmpeg.org/ticket/8492

[54] Formatos de FFmpeg. [En línea]. Disponible en: https://ffmpeg.org/ffmpeg-formats.html

[55] H.264 Baseline vs Main, 21 de febrero de 2012. [En línea]. Disponible en: https://ipvm.com/reports/h264-codec-shootout

[56] Cuál es la diferencia entre los perfiles baseline, main y high, 6 de octubre de 2018. [En línea]. Disponible en: https://support.pelco.com/s/article/What-is-difference-between-baseline-main-and-high-profile-for-the-Sarix-IXE-series-network-IP-cameras-1538586565075

[57] FFmpeg – DirectShow, abril de 2020. [En línea]. Disponible en: https://trac.ffmpeg.org/wiki/DirectShow

[58] Transmisión en vivo MPEG-DASH con Raspberry Pi 3, 3 de julio de 2017. [En línea]. Disponible en: https://www.jungledisk.com/blog/2017/07/03/live-streaming-mpeg-dash-with-raspberry-pi-3/

[59] Explicación del elemento tune en x264, 11 de marzo de 2013. [En línea]. Disponible en: https://superuser.com/questions/564402/explanation-of-x264-tune

[60] Introducción a JSON. [En línea]. Disponible en: http://json.org/json-es.html

[61] Crear un API REST con JSON-server, 27 de noviembre de 2015. [En línea]. Disponible en: https://desarrolloweb.com/articulos/crear-api-rest-json-server.html

[62] API: Qué es y para qué sirve, 23 de agosto de 2019. [En línea]. Disponible en: https://www.xataka.com/basics/api-que-sirve

[63] Qué es una API REST, 25 de abril de 2014. [En línea]. Disponible en: https://desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html

[64] Cómo crear un servidor web en Node.js con el módulo HTTP, 29 de abril de 2020. [En línea]. Disponible en: https://www.digitalocean.com/community/tutorials/how-to-create-a-web-server-in-node-js-with-the-http-module-es

[65] Firebase, Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Firebase

[66] Firebase Hosting. [En línea]. Disponible en: https://firebase.google.com/docs/hosting/

[67] Primeros pasos con Firebase Hosting. [En línea]. Disponible en: https://firebase.google.com/docs/hosting/quickstart

[68] Firebase cli. [En línea]. Disponible en: https://www.xspdf.com/resolution/50492097.html

[69] NetLimiter, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/NetLimiter

[70] DASH-IF, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/DASH-IF

[71] Reproductor de referencia DASH-IF. [En línea]. Disponible en: https://reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html

[72] BOLA: Adaptación de bitrate casi óptima para vídeos online, 18 de junio de 2020. [En línea]. Disponible en: https://arxiv.org/pdf/1601.06748.pdf

[73] Insertar vídeo con reproductor dash.js, 18 de marzo de 2019. [En línea]. Disponible en: https://docs.microsoft.com/es-es/azure/media-services/previous/media-services-embed-mpeg-dash-in-html5

[74] Reproductor Dash.js embebido. [En línea]. Disponible en: https://reference.dashif.org/dash.js/latest/samples/index.html

[75] VLC media player, Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/VLC_media_player

[76] Reproducción de MPEG-DASH en VLC, 10 de agosto de 2015. [En línea]. Disponible en: https://stackoverflow.com/questions/31925434/mpeg-dash-playback-on-vlc

[77] Reproducción MPEG-DASH + HLS nativa en Chrome. [En línea]. Disponible en: https://chrome.google.com/webstore/detail/native-mpeg-dash-%20-hls-pl/cjfbmleiaobegagekpmlhmaadepdeedn?hl=es

[78] Reproducción MPEG-DASH + HLS nativa en Firefox. [En línea]. Disponible en: https://addons.mozilla.org/es/firefox/addon/native-mpeg-dash-hls-playback/

[79] Bitmovin, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Bitmovin

[80] Reproductor Bitmovin embebido, 12 de febrero de 2020. [En línea]. Disponible en: https://bitmovin.com/docs/player/tutorials/get-started-with-the-bitmovin-player

[81] NexStreaming lanza reproductor de streaming NexPlayer HTML5 para navegadores, 15 de septiembre de 2017. [En línea]. Disponible en: https://www.businesswire.com/news/home/20170915005779/es/

[82] NexStreaming, Wikipedia. [En línea]. Disponible en: https://es.qaz.wiki/wiki/NexStreaming

[83] NexPlayer se convierte en socio de soluciones verificadas de Unity. [En línea]. Disponible en: https://www.businesswire.com/news/home/20201210005395/es/

[84] Documentación NexPlayer HTML 5, 14 de diciembre de 2020. [En línea]. Disponible en: https://github.com/NexPlayer/NexPlayer_HTML5_Documentation/blob/master/gettingstarted.md

[85] Microsoft TEAMS, Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Microsoft_Teams

[86] Sincronizar los archivos de SharePoint y Teams con su equipo. [En línea]. Disponible en: https://support.microsoft.com/es-es/office/sincronizar-los-archivos-de-sharepoint-y-teams-con-su-equipo-6de9ede8-5b6e-4503-80b2-6190f3354a88

[87] Visual Studio Code, Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Visual_Studio_Code

[88] PowerShell: Acerca de las directivas de ejecución, 10 de agosto de 2020. [En línea]. Disponible en: https://docs.microsoft.com/es-es/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-7.1

[89] HTTrack, Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/HTTrack

[90] La importancia del GOP, 28 de enero de 2015. [En línea]. Disponible en: https://kvssoft.wordpress.com/2015/01/28/mpeg-dash-gop/

[91] MPEG-DASH bitrate streaming con FFmpeg, 5 de mayo de 2020. [En línea]. Disponible en: https://blog.zazu.berlin/internet-programmierung/mpeg-dash-and-hls-adaptive-bitrate-streaming-with-ffmpeg.html

[92] Elegir la duración óptima de los segmentos. [En línea]. Disponible en: https://streaminglearningcenter.com/blogs/choosing-the-optimal-segment-duration.html

[93] HBO Max: Analizado, 27 de mayo de 2020. [En línea]. Disponible en: https://ottball.com/hbo-max-dissected/

[94] HLS con soporte para Widevine (Disney+). [En línea]. Disponible en: https://github.com/peak3d/inputstream.adaptive/issues/353

[95] ¿Qué es la baja latencia en HLS?, 10 de agosto de 2020. [En línea]. Disponible en: https://bitmovin.com/live-low-latency-hls/

[96] Comparación entre HLS y MPEG-DASH, 17 de diciembre de 2020. [En línea]. Disponible en: https://www.dacast.com/blog/mpeg-dash-vs-hls-what-you-should-know/

[97] NexPlayer admite transmisiones en formato HLS y DASH de baja latencia en Android, iOS, Smart TV, STB, PC y Mac, 16 de enero de 2019. [En línea]. Disponible en: https://www.businesswire.com/news/home/20190116005452/es/

[98] Low Latency DASH (LL-DASH), 24 de septiembre de 2020. [En línea]. Disponible en: https://www.theoplayer.com/blog/low-latency-dash

[99] Lo último en DASH Low Latency. [En línea]. Disponible en: https://dvb.org/wp-content/uploads/2020/03/Latest-on-DASH-low-latency.pdf

[100] H.265, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/High_Efficiency_Video_Coding#:~:text=High%20Efficiency%20Video%20Coding%20(HEVC,MPEG%2D4%20Part%2010)

[101] VP9, Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/VP9

Guía de presentación del Trabajo Fin de Estudios de la Escuela Técnica Superior de Ingeniería Industrial, Informática y de Telecomunicación de la Universidad Pública de Navarra. [En línea]. Disponible en: https://www2.unavarra.es/gesadj/CyD1/ETSIIT/TFE/GuiaPresentacion_TFE_ETSIIIT.pdf

Biblioteca de la Universidad Pública de Navarra. Oficina de Referencia. Guía para citar y referenciar. IEEE Style", 2016. [En línea]. Disponible en: https://www2.unavarra.es/gesadj/servicioBiblioteca/tutoriales/Citar_referenciar_(IEEE).pdf